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 with prover memory footprint almost the same as verifying the circuit in the clear. Most of these protocols can be made to have a *non-interactive* online phase, yielding a *preprocessing NIZK*. In particular, when the preprocessing is realized by a two-round protocol, one obtains a *malicious designated verifier-NIZK (MDV-NIZK)*, which is a recent closely scrutinized variant of DV-NIZK that allows the verifier to generate a public key (first message of two-round protocol) for the prover and a secret key to verify proofs. Though many practically efficient protocols for proving circuit satisfiability over any field are implemented, protocols over the ring $\mathbb{Z}_{2^k}$ are significantly lagging behind, with only a proof-of-concept pioneering work called *Appenzeller to Brie* and a first proposal called *Moz*$\mathbb{Z}_{2^k}$*arella*. 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, 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 protocols over a large 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 basic 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 overwhelming superiority in high security region); for frequently used $40, 80$ bits soundness on $32, 64$-bit CPUs we all offer savings (up to $3\times$ at best).

(2) Through adapting a recently proposed *interactive* authentication code to work over Galois rings, we obtain constant round VOLE-based ZK protocols over $\mathbb{Z}_{2^k}$ with sublinear (in the circuit size) communication complexity, which was previously achieved only over fields.

(3) In order to circumvent the impossibility result of OT-based reusable VOLE, we propose a novel construction of two-round *reusable* VOLE over Galois rings using Galois fields counterpart and powerful tools developed for non-interactive secure computation. We also show that the pseudorandom correlation generator (PCG) approach to extending VOLE without increasing rounds can be generalized to Galois rings. Instantiating a non-interactive version of our basic protocol with a two-round reusable VOLE preprocessing, we obtain MDV-NIZK over $\mathbb{Z}_{2^k}$

with unique features. Such protocols are not only never achieved over rings but also new over small fields.

1 Introduction

A zero-knowledge (ZK) protocol allows a prover to convince a verifier of an assertion (soundness) without revealing any further information beyond the fact that the assertion is true (zero-knowledge). Secure multi-party computation (MPC) allows mutually suspicious players to jointly compute a function of their local inputs without revealing to any corrupted players additional information beyond the output of the function. Though ZK protocols are a strict subset of two-party computation (or simply 2-PC for short), in the context of ZK protocols, there are more stringent efficiency requirements (round complexity, communication complexity, prover/verifier or combined computation complexity). Take the round complexity for example, in the extreme case non-interactive zero knowledge (NIZK), the prover sends a single message, which any verifier can verify using public information available (publicly verifiable), in the idealised random oracle model (ROM) or given a common reference string generated during a setup. In a relaxed variant *designated verifier* NIZK (or DV-NIZK for short) there is a setup phase in which a common reference string (CRS) is generated and a secret verification key is given to the verifier, who is then the designated verifier (cf. [36] and references therein). There is a recent interest in studying a malicious variant of DV-NIZK (MDV-NIZK for short) where the secret key is not generated by a trusted third party during a setup but by the verifier, who at the same time also generates a corresponding public key for the prover. This is a malicious variant in the sense that the verifier can maliciously choose a public key to leak information from proofs generated by the prover and zero-knowledge must hold even in this case. MDV-NIZKs can be interpreted as two-round ZK protocols in the CRS model, where the verifier's first message is *reusable* in the sense that soundness is preserved even the prover knows whether previous proofs are accepted or not. There is an even weaker notion of NIZK in the literature called preprocessing NIZK (PP-NIZK for short), where the setup not only generates a secret verification key for the verifier but also generates a secret proving key for the prover (cf. [34] and reference therein for variants of PP-NIZKs). In this work, we construct more efficient constant round ZK protocols in general. We also make a special effort in minimizing round complexity and reusability of verifier's first message, which leads to MDV-NIZK.

The relation between ZK protocols and secure MPC protocols is one of the most mutually beneficial in cryptography. On the one hand, ZK protocols help transform a *semi-honestly secure* MPC protocol into a *maliciously secure* one [37]. On the other hand, powerful techniques developed for MPC (sometimes even off-the-shelf MPC protocols) contribute greatly to efficient constructions of ZK protocols, especially those that prove arbitrary circuit satisfiability, which is the focus of this work. The MPC-in-the-head paradigm [31] uses an MPC protocol and a commitment scheme to construct a public verifiable NIZK in the

ROM through Fiat-Shamir transform (the prover emulates the evaluation of the circuit in his/her head with imaginary parties using a MPC protocol and commit to the views of these imaginary parties, which the verifier randomly choose a subset to check consistency). The MPC protocol employed could be an *honest-majority* MPC protocol (so at least three parties) which is either semi-honestly or maliciously secure. [31,27,13,1], or even a semi-honestly secure *dishonest-majority* MPC protocol with *preprocessing* (typically uses multiple party version [33,39] with a few exceptions using 2-PC, e.g. [29]; the preprocessing here is different from that in MPC as all imaginary parties are controlled by the prover, who can then use any functionality such as oblivious transfer (OT) channel for free [33]. The bottleneck of this approach is either big proof size e.g. [31,27,13,33], or small proof size but large prover computation time and memory, e.g. [1]. In particular, the large prover memory bottleneck, which is also present in the zero-knowledge succinct non-interactive argument of knowledge (ZK-SNARK, cf [41] and reference therein) paradigm and the interactive oracle proof (IOP) based protocols, imposes a hard limit on the maximum circuit size that a protocol can support in practice. Another well-established MPC related ZK paradigm is the garbled circuit ZK (GCZK) [32,25,47,30]. Unlike the MPC-in-the-head paradigm, here one directly uses a 2-PC protocol (Yao's 2-PC construction [46]), where the prover is playing the role of the sender and the verifier the receiver. This approach yields small prover computation time and memory requirement. But the ZK protocol obtained is inherently interactive (the verifier needs to prepare the garbled circuit) and the communication complexity is huge. Recently, in the quest of scalable ZK protocols that do not have any efficiency bottlenecks discussed above, 2-PC with preprocessing is reconsidered (departing from the MPC-in-the-head [29] and leaning towards GCZK), resulting in a new paradigm of active research in ZK. The starting point is the introduction of the novel idea of *pseudorandom correlation generator (PCG)* [7,9], which is an extension of the pseudorandom generator (PRG) from generating a single copy of randomness to a pair of correlated randomness between mutually distrustful parties. PCG provides a lightweight alternative for the costly preprocessing protocol (now that we do not have functionalities for free as in MPC-in-the-head [33]) at the cost of introducing learning parity with noise (LPN) [6] assumption or its many large field/ring variants. This makes it possible to have the total prover computational overhead (the online phase being non-cryptographic and can be made constant round, even one non-interactive) only slightly bigger than computing the circuit in the clear without any security, and with communication complexity much smaller than GCZK. Another advantage of using a preprocessing 2-PC over GCZK is that the offline phase is statement independent and when assuming a setup phase for preprocessing, one obtains a PP-NIZK. The correlated randomness required here is the (random) vector oblivious linear evaluation (VOLE) correlation, where the sender obtains two random vectors $\mathbf{M}, \mathbf{x}$ over a ring and the receiver obtains a random scalar Δ and a random vector $\mathbf{K}$ such that $\mathbf{M} = \mathbf{K} - \Delta \cdot \mathbf{x}$. This new paradigm of ZK is usually referred to as VOLE-based ZK. In [7], VOLE-based PP-NIZK over large finite fields was first constructed as an application of their

PCG. In [9], PCG for more general correlations including a subfield variant of VOLE correlation (sVOLE for short) was studied, yielding *silent OT extension*, and the previous results on PP-NIZK was carried from proving arithmetic circuits over large fields to Boolean circuits. In [8], a maliciously secure construction of PCG *protocol* for sVOLE correlation was given, which is two-round in the ROM and satisfies an interesting property (useful in particular for constructing non-interactive secure computation NISC) that allows the PCG protocol to be combined with the online protocol, yielding *silent preprocessing* (further pursued in [16], where computationally more efficient PCG protocols are constructed through making structured LPN assumptions). In [22], a more efficient (communicates one finite field element per multiplication gate for a big enough field) online phase technique called line point zero knowledge (LPZK) was proposed that when combining with VOLE protocols yields PP-NIZK in general and MDV-NIZK (with two-round VOLE [8,16,14]) in particular. Concurrent work Wolverine [41] constructed constant round ZK protocols for verifying circuits over arbitrary fields, where in order to further reduce the computational complexity of the underlying sVOLE protocols, the authors slightly modify the PCG idea following [45]. Also concurrent to LPZK and Wolverine, Mac'n'Cheese [5] proposed two different online phase protocols for verifying arbitrary circuits that when used for disjunction have sublinear communications. Follow up works to the above include QuickSilver [43] and then improved LPZK [21], AntMan [42]. QuickSilver uses the idea of LPZK and Wolverine to achieve one field element per multiplication gate for arbitrary field (also, the protocol has sublinear communication for low degree polynomials). Improved LPZK [21] has reduced the communication from 1 field element per multiplication gate to 1/2. AntMan [42] proposed a new authentication coding technique that achieved asymptotic sublinear (in the circuit size) communication for arbitrary circuits. The VOLE-based verification is in fact the information-theoretic message authentication code (IT-MAC) technique in MPC literature and the new technique generalises IT-MAC from authenticating messages to polynomials (hence called IT-PAC, see technical overview for more details).

The models of computation in real-life programming and the computer architectures (such as CPU words) are formulated as operations over the ring $\mathbb{Z}_{2^{32}}$ and $\mathbb{Z}_{2^{64}}$. The fact that half of the ring $\mathbb{Z}_{2^k}$ are zero divisors presents non-trivial technical difficulties for directly performing IT-MAC over them. Naively emulating arithmetic computation over these rings through finite field operations and implementing using one of the above protocols over finite fields incurs significant overhead [26]. Inspired by the techniques developed for MPC (in particular, SPD$\mathbb{Z}_{2^k}$ [17]), Baum et.al. initiated the study of ZK protocols natively supporting computation over $\mathbb{Z}_{2^k}$ [2], where two online phase algorithms were proposed following the finite field counterpart Wolverine [41] and Mac'n'cheese [5], respectively. It was only in a followup work, a first complete proposal Moz$\mathbb{Z}_{2^k}$arella [3] with a QuickSilver style [43,22] protocol together with a PCG VOLE protocol over $\mathbb{Z}_{2^k}$ were implemented. They used the ring extension technique of SPD$\mathbb{Z}_{2^k}$ [17], which incurs sophisticated adjustments (overhead) and more importantly,

the efficiency of such protocols has an inherent undesirable dependency on the security parameter (as the soundness error becomes smaller, the protocol quickly goes inefficient). Given the width and depth of the theoretical study on VOLE-based ZK protocols over finite fields, especially large enough fields, the current state of protocols over application-wise important rings $\mathbb{Z}_{2^k}$ leaves too much to be desired.

Our Contributions.

We introduce more powerful tools for efficient MPC protocols over $\mathbb{Z}_{2^k}$ into the VOLE-based construction of constant-round ZK protocols. On top of optimizing the communication complexity and computation complexity (including memory footprint and combined prover-verifier computation complexity), we also pay special attention to achieving the minimum round complexity with the strongest form of malicious security. We obtain three types of results.

(1) Targeting the state-of-the-art ZK protocol over $\mathbb{Z}_{2^k}$, Moz$\mathbb{Z}_{2^k}$arella, which in turn uses QuickSilver as blueprint, we propose a competing basic protocol (including the PCG-style sublinear VOLE protocol for implementing the protocol). Both protocols have similar impressively low computational overhead and linear communication complexity, while the efficiency of our basic protocol does not have the undesirable dependence on security parameter (see Theorem 2). This means that in all high security region applications, our basic protocol has overwhelming advantage over Moz$\mathbb{Z}_{2^k}$arella. We then compare concrete performance between the two frequently used 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 comparison against Moz$\mathbb{Z}_{2^k}$arella (online phase comparison). "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.[1]

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

(2) Targeting the sublinear (in the circuit size) communication ZK protocol over fields, AntMan, we construct the first protocol over $\mathbb{Z}_{2^k}$ with the same asymptotic efficiency. (We do not see any possibility that similar efficiency can be achieved using the Moz$\mathbb{Z}_{2^k}$arella approach.) Similar to the results over fields,

[1] We select RMFEs over binary field according to [11], and lift to Galois rings via the approach in [18].

we need a large circuit size (estimated at least 2^{20}) to allow the computational cost of setting up the more efficient authentication coding scheme be averaged out. Assuming proving satisfiability of a SIMD circuit for m copies of data, the sublinear communication protocol has better efficiency as m increases. For example, for $\kappa = 40$, we use $(16, 45)$-RMFEs. We estimate that the sublinear communication protocol starts to outperform our basic protocol when computing a SIMD circuit over $\mathbb{Z}_{2^{32}}$ for more than 16×12 copies of data [2].

(3) Targeting minimum round complexity and the strongest malicious security, we start with a variant of our basic protocol with *non-interactive* online phase and combine it with a two-round reusable VOLE protocol over Galois rings. We propose a novel construction of reusable VOLE over Galois rings using reusable VOLE over Galois fields and powerful tools from the study of reusable non-interactive secure computation (NISC). We further show that the PCG approach to extending VOLE without increasing rounds can be generalized to Galois rings. Putting all these together, we obtain a MDV-NIZK with the kind of communication complexity and computation complexity unique for VOLE-based ZK protocols. On the other hand, unlike the conventional MDV-NIZKs, our MDV-NIZK protocol uses a random oracle and this may be the reason for its unusually good parameters.

Technical Overview

We begin with a basic construction that illustrates the high-level idea behind both of our constructions in Section 3: constructing a ZK protocol over a Galois ring $\mathtt{GR}(2^k, d)$ (assuming a PCG-style VOLE protocol over the Galois ring, which we also construct in Section 4) and then converting it into $\mathbb{Z}_{2^k}$ efficiently using amortisation techniques. The reason for separating our basic construction from the sublinear communication variant is that although the techniques applied in the latter share high-level similarity with the former, there are substantial low-level differences that, in particular, make the protocols obtained from the latter do not have a non-interactive online phase. Building a ZK protocol over Galois ring $\mathtt{GR}(2^k, d)$ following the blueprint of a ZK protocol over a Galois field is straightforward. For the basic protocol, we follow the QuickSilver style [43,22] protocol, same as Moz$\mathbb{Z}_{2^k}$arella does. By analogy between Galois fields and Galois rings, the security parameter is now measured by 2^{-d}. But as the relation we wish to prove is over $\mathbb{Z}_{2^k}$ (so are the witnesses), we are paying d times communication for the Galois structure. In order to amortise this non-trivial cost, we manage to make one evaluation over the Galois ring computes m copies of data over $\mathbb{Z}_{2^k}$, maintaining a communication cost independent of the security parameter (the ratio $\frac{d}{m}$ is upper bounded by a constant). Amortisation techniques are well studied in the MPC literature, mostly for constructing efficient protocols over small size fields (e.g. binary) such as the coding approach MiniMAC [19] and the versatile Reversed Multiplication Friendly Embedding (RMFE) [11,12], which is also playing an important role in MPC protocols over $\mathbb{Z}_{2^k}$ [23]. We take this opportunity to note a subtle difference between the MPC literature and the

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

VOLE-based ZK literature: efficient VOLE-based ZK protocols over small fields can be directly obtained from sVOLE without resorting to amortisation techniques (another evidence that these two problems are rather different, although we may be using the same set of techniques on them). We are indeed inspired by the recent Galois ring RMFE techniques developed for dishonest-majority MPC [23], which unfortunately does not work well with the PCG-style VOLE protocol. We then formulate the required functionality and construct a protocol for the functionality that is compatible with PCG-style VOLE protocol. Finally, same as all above mentioned amortisation techniques, our ZK protocol (stated for an SIMD circuit) is to be combined with a circuit rearrangement technique that transforms a generic circuit into a SIMD circuit that has equivalent functionality. The transformation incurs an overhead that depends on the topology of the given circuit. According to [19,12], a circuit is called *well-formed* (with respect to an integer m) if every layer of the circuit is $\Omega(m)$ gates wide and the number of output values that are inputs to subsequent layers is either 0 or $\Omega(m)$. The transformation overhead for a well-formed circuit (into a SIMD circuit with m copies of data) is very small. In our application scenario of scalable ZK protocols, we generally consider large circuits and $m = 16$ for $\mathbb{Z}_{2^{32}}$ ($m = 27$ for $\mathbb{Z}_{2^{64}}$). Moreover, circuits coming from real-life applications are usually more structured than a random circuit.

We now proceed with our sublinear communication variant construction. Consider a SIMD circuit with $m = m_1 \times m_2$ copies of data over $\mathbb{Z}_{2^k}$ and we will use RMFE amortisation to map m_1 copies of data over $\mathbb{Z}_{2^k}$ into one copy of data over $\mathtt{GR}(2^k, d)$, followed by applying a new amortisation technique that operates on a batch of m_2 elements in $\mathtt{GR}(2^k, d)$. Before we continue, let us point out a fact that is important for distinguishing the previous RMFE amortisation technique from the new technique that enables sublinear communication. Note that RMFE amortisation does not improve communication efficiency (in fact, it incurs a constant fraction of degradation as we always have $m_1 < d$). In practice, we choose the smallest m_1 such that a good (m_1, d)-RMFE with small $\frac{d}{m_1}$ exist and 2^{-d} is less than the required soundness error. The new amortisation technique authenticates m_2 values over $\mathtt{GR}(2^k, d)$ in a batch using communication cost less than authenticating them naively one-by-one, hence the sublinear communication. We generalise the Information-Theoretic Polynomial Authentication Code (IT-PAC) technique of [42] from Galois fields to Galois rings, which now involves interpolating a polynomial over $\mathtt{GR}(2^k, d)$ using m_2 elements in $\mathtt{GR}(2^k, d)$ and authenticating the polynomial through checking the IT-MAC of its random evaluation. We note here that the nice structures of Galois rings are crucial for the whole thing to work. For example, the ring $\mathbb{Z}_{2^{k+s}}$ used in Moz$\mathbb{Z}_{2^k}$arella [3] will not support the Lagrange interpolation step, which requires the ring to have a large exceptional set. The IT-PAC techniques help reduce the communication complexity by m_2 in this case. Different from m_1 (determined by security parameter), it is desirable that m_2 should be chosen as big as possible. Another trick of reducing verifying a generic circuit to verifying a SIMD circuit was proposed in [42] 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 generation of IT-PAC increases the computational complexity considerably and moreover it needs interaction.

Finally, in order to achieve the strongest malicious security, we need an efficient protocol that generates reusable VOLE correlation. According to [14], reusable VOLE can not be constructed solely based on OT, which rules out the conventional OT-based construction. For example, the base VOLE construction in [41,43] is not reusable. Exploiting the decomposition of the Galois ring $\mathtt{GR}(2^k, d)$ with respect to its subfield $\mathbb{F}_{2^d}$ (another nice structure of Galois rings), we propose a novel idea of constructing reusable VOLE over $\mathtt{GR}(2^k, d)$ using parallel calls to a reusable VOLE over $\mathbb{F}_{2^d}$. The decomposition structure also helps in making this step more efficient than the conventional OT-based construction (intuitively, emulating the computation over $\mathtt{GR}(2^k, d)$ using Boolean computation). The challenge in making this idea work lies in how to prevent a malicious sender from providing inconsistent VOLE inputs across parallel calls. For this we use an off-the-shelf protocol developed for constructing NISC called eVOLE [14,22], which allows the sender to prove to the receiver that certain coefficients of different inputs in parallel calls are equal. With the efficiency saving from the decomposition structure, the extra cost of the eVOLE is canceled out and we basically obtain reusability for free. To generate reusable VOLE correlation more efficiently, we show that the PCG-style VOLE extension can be extended to Galois rings. These PCG protocols are better than the PCG protocols over $\mathbb{Z}_{2^k}$ in two aspects. First of all, the LPN assumption over $\mathtt{GR}(2^k, d)$ is more secure. We show a reduction relating to LPN over the field $\mathbb{F}_{2^d}$. The same reduction severely affects the security of an earlier version of Moz$\mathbb{Z}_{2^k}$arella as reported in [35] and, to evade this attack, the LPN error vector should contain only components that are units in $\mathbb{Z}_{2^k}$. Our results show that we can either go without this modification and tolerating a slightly degraded security, or, adapt a similar modification while remaining on safer ground than LPN over $\mathbb{Z}_{2^k}$, as the fraction of units in $\mathtt{GR}(2^k, d)$ is close to one. Secondly, the Galois structure makes it natural to seamlessly generalise different PCG protocols over Galois fields to PCG protocols over Galois rings with all their different features well-preserved. We include the generalisation of the two most important PCG protocol constructions, one based on primal LPN with optimised computational complexity [45] and one based on dual LPN with optimal round complexity [8].

Related Works.

The previous works most relevant to this work is Moz$\mathbb{Z}_{2^k}$arella [3]. which still has advantage over our basic construction on proving contrived circuits that incur heavy overhead in rearranging into SIMD circuits. Other than that, our basic construction gives better protocols. Moreover, Moz$\mathbb{Z}_{2^k}$arella does not have sublinear communication variant and MDV-NIZK variant. Sublinear communication ZK protocols with very attractive communication complexity for very specific relations have been constructed over finite fields [5,43]. There is an orthogonal line of active research on constructing different variants of DV-NIZKs including its dual model designated prover NIZK (DP-NIZK) from standard assumptions

such as learning with error (LWE) (see [36,34] and references therein). Our MDV-NIZK uses multiple computational assumptions coming from many of its building blocks, and moreover, we use random oracle, which makes it incomparable to this line of works. Finally, we notice that an efficient threshold designated-verifier Zero-Knowledge proofs over finite fields called Feta [4] and similar ideas [44] were proposed to overcome the designated verifier restriction. We believe similar improvements as above can be studied over $\mathbb{Z}_{2^k}$. Due to the space limit, we only discuss generic circuit solutions and restrict to designated verifier ZK in this work.

2 Preliminaries

Notations. In this paper, bold letters (e.g. $\boldsymbol{a}, \boldsymbol{b}$) are used to denote vectors. Besides, we use x_i to denote the i_{th}-component of the vector $\boldsymbol{x}$. 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 in Section 3. We also use $\boldsymbol{x}[a : b]$ to denote the set $\{x_i \mid i \in [a, b]\}$. We use $x \xleftarrow{\$} \mathcal{R}$ to denote that x is uniformly sampled from a ring $\mathcal{R}$ and denote the uniform distribution over $\mathcal{R}$ by $\mathrm{U}_{\mathcal{R}}$. For a map $\phi : \mathcal{R}_1 \mapsto \mathcal{R}_2$, we naturally extend it to be defined over vector space $\mathcal{R}_1^n$ and matrix space $\mathcal{R}_1^{m \times n}$.

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 $\mathbb{F}_p$. A Galois ring over $\mathbb{Z}_{p^k}$ of degree d denoted by $\mathtt{GR}(p^k, d)$ is a ring extension $\mathbb{Z}_{p^k}[X]/(f(X))$ of $\mathbb{Z}_{p^k}$. The readers may refer to the text [40] for a friendly exposition. Same as Galois fields, there is a bound on the number of roots.

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

Lemma 1 immediately gives that for any nonzero degree-r polynomial $f(x)$ over $\mathtt{GR}(p^k, d)$, we have that

$$\Pr\Big[f(\alpha) = 0 \;\Big|\; \alpha \xleftarrow{\$} \mathtt{GR}(p^k, d)\Big] \leq rp^{-d}.$$

In particular, we have that $1/p^d$ fraction of elements are zero divisors in $\mathtt{GR}(p^k, d)$, i.e. $(1 - 1/p^d)$ fraction of elements are invertible.

VOLE based MAC. (Random) Vector OLE (VOLE) is a two-party 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 random vectors $\boldsymbol{M}, \boldsymbol{x}$, while the receiver $P_{\mathcal{R}}$ obtains a random scalar Δ and a random vector $\boldsymbol{K}$ such that $\boldsymbol{M} = \boldsymbol{K} - \Delta \cdot \boldsymbol{x}$ holds. We formalize the VOLE ideal functionality over Galois ring $\mathtt{GR}(2^k, d)$ following the PCG-style formulation in Figure 1.

The above VOLE correlations can be viewed as Message Authentication Codes (MACs) that authenticate $\boldsymbol{x}$, denoted by $[\boldsymbol{x}]$. We then call $\boldsymbol{M}$ the MAC tags, $\boldsymbol{K}$ the local keys and Δ the global key. It can be easily observed that this kind of

MAC is linearly homomorphic. Suppose $[x_1], ..., [x_l]$ have been already distributed and the coefficients $c, c_1, ..., c_l \in \mathtt{GR}(2^k, d)$ are public, the two parties can obtain $[y]$ without interaction, where $y = c + \sum_{i\in[l]} c_i \cdot x_i$, by setting $M_y := \sum_{i\in[l]} M_{x_i}$ and $K_y := \Delta \cdot c + \sum_{i\in[l]} c_i \cdot K_{x_i}$, respectively. According to this observation, $P_{\mathcal{S}}$ and $P_{\mathcal{R}}$ can obtain $[y]$ from a random MAC $[x]$ by sending $y - x$ to $P_{\mathcal{R}}$ while maintaining the privacy of y.

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

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

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

(1) If $P_{\mathcal{R}}$ is honest, sample $\boldsymbol{K} \xleftarrow{\$} \mathtt{GR}(2^k, d)^n$. Otherwise receive $\boldsymbol{K}$ from $\mathcal{A}$.

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

(3) Send $(\boldsymbol{x}, \boldsymbol{M})$ to $P_{\mathcal{S}}$ and $\boldsymbol{K}$ to $P_{\mathcal{R}}$.

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

Reverse Multiplicative Friendly Embedding. Reverse Multiplicative Friendly Embedding (RMFE for short) was first introduced in [11], which packs multiple multiplications over a field $\mathbb{F}_q$ to one multiplication over its extension $\mathbb{F}_{q^d}$. It was further showed in [18] that RMFEs over finite fields can be lifted to Galois rings.

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

$$\boldsymbol{x} * \boldsymbol{y} = \psi(\phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{y}))$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \mathtt{GR}(p^k, r)^m$.

RMFEs have the following properties, whose proofs can be found in Section A in Supplementary Materials.

Lemma 2. *Let (ϕ, ψ) be a (m, d)-RMFE over Galois ring $\mathtt{GR}(p^k, r)$, we have that $\mathtt{GR}(p^k, rd) = \phi(\mathbf{1}) \cdot \mathrm{Im}(\phi) \oplus \mathrm{Ker}(\psi)$.*

Lemma 3. *There always exists a (m, d)-RMFE over Galois ring $\mathtt{GR}(p^k, r)$ with $\phi(\mathbf{1}) = 1$.*

RMFEs with constant rate (the ratio $\frac{m}{d}$) exist over arbitrary Galois rings.

Lemma 4 ([18]). *There exists a family of (m, d)-RMFEs over Galois ring $\mathtt{GR}(p^k, r)$, where $d = \mathcal{O}(m)$.*

Let $r = 1, p = 2$, we immediately obtain RMFEs from $\mathbb{Z}_{2^k}^m$ to $\mathtt{GR}(2^k, d)$ and w.l.o.g. we assume $\phi(\mathbf{1}) = 1$. For the rest of this paper, we always consider such RMFEs. In particular, as showed in [11], there exists a family of (m, d)-RMFEs over $\mathbb{Z}_{2^k}$ with

$$\lim_{m\to\infty} \frac{d}{m} = 4.92.$$

Zero-Knowledge Proof. The zero-knowledge proof functionality ($\mathcal{F}_{\mathrm{ZK}}^m$, Figure 12) for circuit satisfiability allows prover $\mathcal{P}$ to prove knowledge of a witness $\boldsymbol{w}$ for some public circuit $\mathcal{C}$, i.e. $\mathcal{C}(\boldsymbol{w}) = 1$ without revealing any additional information to verifier $\mathcal{V}$. We generalize the functionality by allowing $\mathcal{P}$ to prove multiple of witnesses of the same $\mathcal{C}$, simultaneously.
Equality Test & Oblivious Transfer. The equality test functionality ($\mathcal{F}_{\mathrm{EQ}}$, Figure 13) allows two parties $\mathcal{P}$ and $\mathcal{V}$ to check their inputs are equivalent with $\mathcal{P}$'s input revealed to $\mathcal{V}$. We define the oblivious transfer functionality ($\mathcal{F}_{\mathrm{OT}}$, Figure 14), which receives a bit from $P_{\mathcal{S}}$ and two strings from $P_{\mathcal{R}}$, then sends one of the strings to $P_{\mathcal{S}}$ according to the bit. See more preliminaries in Section A in Supplementary Material.

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

We begin with describing our basic construction, as a warm-up for the more sophisticated variant with sublinear communication online phase.

3.1 Basic Construction

Constructing a ZK protocol over a Galois ring $\mathtt{GR}(2^k, d)$ following the blueprint of a well-worked-out protocol over a Galois field is straightforward. The technicality of our construction lies in the second step, which is converting the protocol efficiently into one over $\mathbb{Z}_{2^k}$ using an RMFE (ϕ, ψ). The challenge lies in that, by definition, multiplication is only preserved for two elements in $\mathtt{GR}(2^k, d)$ that lie in the image of ϕ, while in the ZK protocol over $\mathtt{GR}(2^k, d)$, elements of $\mathtt{GR}(2^k, d)$ are multiplied without knowing whether they are belonging to the image of ϕ. We note that there are off-the-shelf VOLE-based solutions in the MPC literature for enforcing that multiplicands are belonging to image of ϕ [23,12]. But for ZK protocols, we also need the solution to work with the PCG-style sublinear VOLE protocol. Being the first to use RMFE on preprocessing of VOLE-based ZK protocols, we define the functionality required and construct it before describing how it is used in our ZK protocol over $\mathtt{GR}(2^k, d)$.

Let $\phi : \mathbb{Z}_{2^k}^m \mapsto \mathtt{GR}(2^k, d)$ and $\psi : \mathtt{GR}(2^k, d) \mapsto \mathbb{Z}_{2^k}^m$ be a (m, d)-RMFE pair over $\mathbb{Z}_{2^k}$. As ϕ, ψ are $\mathbb{Z}_{2^k}$-linear, their composition $\phi \circ \psi : \mathtt{GR}(2^k, d) \mapsto \mathtt{GR}(2^k, d)$, denoted by τ, is a $\mathbb{Z}_{2^k}$-linear map as well. We use the VOLE based MAC to authenticate values in $\mathtt{GR}(2^k, d)$. To be compatible with RMFEs, we formulate the

ideal functionality $\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$ in Figure 2 that extends $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$ by additionally allowing the two parties to obtain $([x],[\tau(x)])$, where $x \xleftarrow{\$} \mathtt{GR}(2^k,d)$. We say $([x],[\tau(x)])$ is the re-embedding pair MAC of x.

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

$\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$ extends the existing VOLE functionality $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$ (Figure 1). **Init** and **Extend** are identical to those in $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$, respectively. Let (ϕ,ψ) be a (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 $\boldsymbol{x}, \boldsymbol{M} \xleftarrow{\$} \mathtt{GR}(2^k,d)^n$; otherwise, receive $\boldsymbol{x}, \boldsymbol{M} \in \mathtt{GR}(2^k,d)^n$ from $\mathcal{A}$. Compute $\boldsymbol{\eta} := \tau(\boldsymbol{x}) - \boldsymbol{x} \in \mathtt{Ker}(\psi)^n$ and send $\boldsymbol{\eta}$ to $P_{\mathcal{R}}$.

(2) If $P_{\mathcal{R}}$ is honest, compute $\boldsymbol{K} := \boldsymbol{M} + \Delta \cdot \boldsymbol{x} \in \mathtt{GR}(2^k,d)^n$; otherwise, receive $\boldsymbol{K} \in \mathtt{GR}(2^k,d)^n$ from $\mathcal{A}$ and recompute $\boldsymbol{M} := \boldsymbol{K} - \Delta \cdot \boldsymbol{x}$.

(3) Send $(\boldsymbol{x}, \boldsymbol{M})$ to $P_{\mathcal{S}}$ and $(\boldsymbol{K}, \boldsymbol{\eta})$ to $P_{\mathcal{R}}$.

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

Our protocol for $\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$ crucially relies on the key observation that $\mathtt{GR}(2^k,d)$ is the direct sum of $\mathtt{Ker}(\psi)$ and $\mathtt{Im}(\phi)$, and the SIMD witnesses over $\mathbb{Z}_{2^k}$ one-to-one correspond to a vector over $\mathtt{Im}(\phi)$ (we use ϕ to map the witnesses to a vector over $\mathtt{GR}(2^k,d)$). Thus, we allow the projection of $\boldsymbol{x}$ on $\mathtt{Ker}(\psi)$ to be revealed to $\mathcal{V}$, which enables the two parties to obtain the re-embedding pair MACs of $\boldsymbol{x}$. So far, the prover can cheat during the generation of these re-embedding pair MACs. We then have the two parties generate $n+s$ re-embedding pair MACs and sacrifice the extra s re-embedding pair MACs 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 big enough (e.g. $s = d = \kappa + 1$). We give the protocol $\Pi_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$ in Figure 3, and we have the following theorem, whose security proof can be found in in Section C.1 in Supplementary Material. To obtain n re-embedding pair MACs, our approach requires $n+s$ VOLE correlations with communication of $(n+3s)$ Galois ring elements and ns coefficients over $\mathbb{Z}_{2^k}$, i.e. 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 $\mathcal{O}(1)$.

Theorem 1. *$\Pi_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$ UC-realizes $\mathcal{F}_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$ in the $(\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}, \mathcal{F}_{\text{EQ}})$-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 $2^{-s}+2^{-d}$.*

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

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

Extend-pair:

1. **Construct:**

(a) Both parties send $(\mathtt{Extend}, n+s)$ to $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$. $P_{\mathcal{S}}$ receives $\boldsymbol{M}, \boldsymbol{x} \in \mathtt{GR}(2^k,d)^{n+s}$, and $P_{\mathcal{R}}$ receives $\boldsymbol{K} \in \mathtt{GR}(2^k,d)^{n+s}$, such that $\boldsymbol{M} = \boldsymbol{K} - \Delta \cdot \boldsymbol{x}$ holds. Thus, the parties now obtain $[x_i], i \in [n+s]$.

(b) $P_{\mathcal{S}}$ computes $\boldsymbol{\eta} := \tau(\boldsymbol{x}) - \boldsymbol{x}$, then sends $\boldsymbol{\eta} \in \mathtt{Ker}(\psi)^{n+s}$ to $P_{\mathcal{R}}$. If $\boldsymbol{\eta} \notin \mathtt{Ker}(\psi)^{n+s}$, $P_{\mathcal{R}}$ aborts.

(c) $P_{\mathcal{S}}$ sets $\boldsymbol{M}' := \boldsymbol{M}$, and $P_{\mathcal{R}}$ sets $\boldsymbol{K}' := \boldsymbol{K} + \Delta \cdot \boldsymbol{\eta}$. Note that $\boldsymbol{M}' = \boldsymbol{K}' - \Delta \cdot \tau(\boldsymbol{x})$ holds, so the parties now obtain $[\tau(x_i)], i \in [n+s]$.

2. **Sacrifice:**

(a) $P_{\mathcal{R}}$ samples $\boldsymbol{\chi}^1, ..., \boldsymbol{\chi}^s \xleftarrow{\$} \mathbb{Z}_{2^k}^n$, and sends them to $P_{\mathcal{S}}$.

(b) For $i \in [s]$, $P_{\mathcal{S}}$ computes $a_i = x_{n+i} + \sum_{j\in[n]} \chi_j^i \cdot x_j$ and $b_i = \tau(x_{n+i}) + \sum_{j\in[n]} \chi_j^i \cdot \tau(x_j)$. $P_{\mathcal{S}}$ sends $\boldsymbol{a}, \boldsymbol{b}$ to $P_{\mathcal{R}}$. $P_{\mathcal{S}}$ computes $\hat{M}_i := M_{n+i} + \sum_{j\in[n]} \chi_j^i \cdot M_j$, for $i \in [s]$.

(c) $P_{\mathcal{R}}$ checks $b_i - a_i = \eta_{n+i} + \sum_{j\in[n]} \chi_j^i \cdot \eta_j$, for $i \in [s]$ and $\boldsymbol{b} \in \mathtt{Im}(\phi)^s$. If the check fails, $P_{\mathcal{R}}$ aborts. $P_{\mathcal{R}}$ computes $\hat{M}_i' := K_{n+i} + \sum_{j\in[n]} \chi_j^i \cdot K_j - \Delta \cdot a_i$, for $i \in [s]$.

(d) $P_{\mathcal{R}}$ sends $\hat{\boldsymbol{M}}'$ to $\mathcal{F}_{\text{EQ}}$ as $\mathcal{V}$. $P_{\mathcal{S}}$ sends $\hat{\boldsymbol{M}}$ to $\mathcal{F}_{\text{EQ}}$ as $\mathcal{P}$. $P_{\mathcal{R}}$ receives $\hat{\boldsymbol{M}}$. $P_{\mathcal{S}}$ and $P_{\mathcal{R}}$ abort if the equality test fails.

3. **Output:** Output $([x_i], [\tau(x_i)])$ for $i \in [n]$.

Fig. 3: Protocol for authenticating re-embedding pairs over $\mathtt{GR}(2^k,d)$ in the $(\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}, \mathcal{F}_{\text{EQ}})$-hybrid model.

Now we give the basic zero-knowledge proof protocol in Figure 4. Suppose the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ have agreed on a SIMD circuit $\mathcal{C}$ over $\mathbb{Z}_{2^k}$ with n inputs and t multiplication gates, and $\mathcal{P}$ 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 $\boldsymbol{\omega} := \phi(\boldsymbol{w}_1, ..., \boldsymbol{w}_m)$, and sends $\boldsymbol{\delta} := \boldsymbol{\omega} - \boldsymbol{\mu}$ to $\mathcal{V}$. They can obtain $[\tau(\boldsymbol{\omega})] := [\tau(\boldsymbol{\mu})] + \tau(\boldsymbol{\delta})$. Note that if $\boldsymbol{\omega} \in \mathtt{Im}(\phi)^n$, 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. While for `Mul` gates, $\mathcal{P}$ is required to send the outputs with masks. As RMFE can only preserve one multiplication, all values on the circuit wires should be ensured in the image of ϕ, which can be achieved with the help of re-embedding pair MACs. More specifically, for the i-th multiplication gate in $\mathcal{C}$ with inputs $\omega_\alpha, \omega_\beta$, if d_i sent to $\mathcal{V}$ is equal to $\omega_\alpha \cdot \omega_\beta - \nu_i$, they can compute $[\omega_\gamma] := [\tau(\omega_\alpha \cdot \omega_\beta)] = [\tau(\nu_i)] + \tau(d_i)$. To verify that each d_i is computed honestly, the two parties additionally compute $[\hat{\omega}_\gamma] := [\omega_\alpha \cdot \omega_\beta] = [\nu_i] + d_i$. One can observe that

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

holds if d_i is correct. Therefore, it can be used to detect malicious behaviors by letting $\mathcal{P}$ send $A_{0,i} := M_{\omega_\alpha} \cdot M_{\omega_\beta}, A_{1,i} := \omega_\alpha \cdot M_{\omega_\beta} + \omega_\beta \cdot M_{\omega_\alpha} - M_{\hat{\omega}_\gamma}$ to $\mathcal{V}$. Further, we 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 M_π, π, respectively. In fact, as $\mathtt{GR}(2^k,d)$ contains a subfield $\mathbb{F}_{2^d}$, $\boldsymbol{\chi}$ can be sampled from $\mathbb{F}_{2^d}^t$. Finally, for the output wire ω_h, if both parties follow the protocol honestly, the equation $K_{\omega_h} = M_{\omega_h} + \Delta$ should hold. Thus, we let $\mathcal{P}$ open $[\omega_h]$ by sending M_{ω_h} to $\mathcal{V}$. We have the following theorem, whose security proof can be found in Section C.2 in Supplementary Material. The protocol $\Pi_{\text{ZK}}^{m,n,t}$ transfers one $\mathtt{GR}(2^k,d)$ element and one random coefficient over $\mathbb{F}_{2^d}$ per multiplication gate, yielding amortized communication complexity $(kd+d)/m$ bits, which is independent of the statistical security parameter κ as d/m is constant.

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

Note that this basic construction has constant rounds (two rounds for both $\Pi_{\text{embVOLE}}^{\mathtt{GR}(2^k,d)}$ and $\Pi_{\text{ZK}}^{m,n,t}$). By Fiat-Shamir heuristic, the two protocols can be made non-interactive, respectively, and the overall round complexity of the basic construction is one. Thus, one can obtain a MDV-NIZK protocol in ROM if instantiating with a two-round reusable VOLE protocol.

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 $\boldsymbol{w}_i \in \mathbb{Z}_{2^k}^n$ such that $\mathcal{C}(\boldsymbol{w}_i) = 1$, $i \in [m]$.

Offline phase

1. $\mathcal{P}$ and $\mathcal{V}$ send $(\texttt{Init})$ to $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$, and $\mathcal{V}$ receives $\Delta \in \mathtt{GR}(2^k, d)$.

2. $\mathcal{P}$ and $\mathcal{V}$ send $(\texttt{Extend-pair}, n+t)$ to $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$, which returns authenticated pairs $([\mu_i], [\tau(\mu_i)])_{i\in[n]}$, $([\nu_j], [\tau(\nu_j)])_{j\in[t]}$, where all μ_i, ν_j are sampled uniformly at random in $\mathtt{GR}(2^k, d)$. Note that $\mathcal{V}$ also learns $\tau(\boldsymbol{\mu}) - \boldsymbol{\mu}$ and $\tau(\boldsymbol{\nu}) - \boldsymbol{\nu}$ from $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$.

3. $\mathcal{P}$ and $\mathcal{V}$ send $(\texttt{Extend}, 1)$ to $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$, which returns authenticated value $[\pi]$, where π is sampled uniformly at random in $\mathtt{GR}(2^k, d)$.

Online phase

1. For input $W = (\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_m)$, $\mathcal{P}$ 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 $K_{\omega_j} = M_{\omega_j} + \Delta \cdot \omega_j$ for $j \in \{\alpha, \beta\}$, and $K_{\hat{\omega}_\gamma} = M_{\hat{\omega}_\gamma} + \Delta \cdot \hat{\omega}_\gamma$.

   - $\mathcal{P}$ computes $A_{0,i} := M_{\omega_\alpha} \cdot M_{\omega_\beta} \in \mathtt{GR}(2^k, d)$ and $A_{1,i} := \omega_\alpha \cdot M_{\omega_\beta} + \omega_\beta \cdot M_{\omega_\alpha} - M_{\hat{\omega}_\gamma} \in \mathtt{GR}(2^k, d)$.

   - $\mathcal{V}$ computes $B_i := K_{\omega_\alpha} \cdot K_{\omega_\beta} - \Delta \cdot K_{\hat{\omega}_\gamma} \in \mathtt{GR}(2^k, d)$.

4. $\mathcal{P}$ and $\mathcal{V}$ do the following check:

   (a) $\mathcal{P}$ sets $A_0^* := M_\pi, A_1^* := \pi$, and $\mathcal{V}$ sets $B^* := K_\pi$ so that $B^* = A_0^* + \Delta \cdot A_1^*$.

   (b) $\mathcal{V}$ draws a uniformly random $\boldsymbol{\chi}$ from $\mathbb{F}_{2^d}^t$ and sends it to $\mathcal{P}$.

   (c) $\mathcal{P}$ computes $X := \sum_{i\in[t]} \chi_i \cdot A_{0,i} + A_0^* \in \mathtt{GR}(2^k, d)$ and $Y := \sum_{i\in[t]} \chi_i \cdot A_{1,i} + A_1^* \in \mathtt{GR}(2^k, d)$, and sends (X, Y) to $\mathcal{V}$.

   (d) $\mathcal{V}$ computes $Z := \sum_{i\in[t]} \chi_i \cdot B_i + B^* \in \mathtt{GR}(2^k, d)$, and checks whether $Z = X + \Delta \cdot Y$ holds. If the check fails, $\mathcal{V}$ outputs `false` and aborts.

5. For the single output wire ω_h, both parties hold $[\omega_h]$.

   - $\mathcal{P}$ sends M_{ω_h} to $\mathcal{V}$.

   - $\mathcal{V}$ checks whether $K_{\omega_h} = M_{\omega_h} + \Delta \cdot \phi(\mathbf{1})$. If the check fails, $\mathcal{V}$ outputs `false`. Otherwise, $\mathcal{V}$ outputs `true`.

Fig. 4: Zero-knowledge protocol for circuit satisfiability over $\mathbb{Z}_{2^k}$ in the $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$-hybrid model.

3.2 Sublinear Communication Variant

The RMFE amortisation techniques in the basic construction only reduces the cost of performing the proof over a Galois ring $\mathtt{GR}(2^k, d)$ (has a sufficiently large fraction of units) instead of the target ring $\mathbb{Z}_{2^k}$ (smaller but with half units). The communication complexity is slightly bigger than if we were to perform the proof directly over $\mathbb{Z}_{2^k}$ (assume $\mathbb{Z}_{2^k}$ had many units as $\mathbb{F}_{2^k}$). In order to reduce the communication complexity, we need to use a different amortisation technique on top of it. The main idea is to use the Polynomial Authentication Code (PAC) instead of the MAC, which allows to authenticate a batch of m_2 values simultaneously. PAC is a generalisation of 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 $M = K - \Delta \cdot f(\Lambda)$. The PAC for $f(\cdot)$, denoted by $[f(\cdot)]$, can be viewed as a MAC for $f(\Lambda)$. In particular, the polynomial $f(\cdot)$ to be authenticated is uniquely determined by the values needed to be authenticated via Lagrange interpolation, if the evaluation points are picked appropriately. In a high level, the protocol starts with $\mathcal{P}$ and $\mathcal{V}$ obtaining PACs for the SIMD witnesses, then they evaluate the circuit using the 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 summarise 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 a (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 the 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. In [42], an efficient interactive protocol that generates a batch of PACs over Galois fields, using an Additively Homomorphic Encryption (AHE) scheme was proposed. We generalise this PAC construction to work over Galois rings (including the AHE, see C.3), denoted by Π_{PAC} (see Figure 21 due to space constraint). 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 evaluation is completed, $\mathcal{P}$ and $\mathcal{V}$ do a series of checks. The first check is for the τ-consistency. We adapt the `BatchCheck` procedure in [42] to allow the verifier to check two sets of polynomials additionally satisfy the τ-consistency simultaneously. 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 set of polynomials. To avoid potential information leakage from linear combinations,

the opening polynomials are masked with a pair of random polynomials that satisfy the degree requirement and τ-consistency. Besides, $\mathcal{V}$ should be convinced that the opened polynomials are consistent to their PACs. The PAC amortisation techniques allow the polynomial key Λ to be opened to $\mathcal{P}$, after all polynomials needed to be authenticated are authenticated. Then the PACs are turned to MACs for the polynomial evaluations on Λ naturally, and $\mathcal{V}$ can check this by $\mathcal{P}$ opening the MACs.

The next check is for multiplication. Since the PACs are turned to MACs naturally after the polynomial key is opened, we can use the multiplication check procedure of $\Pi_{\text{ZK}}^{m,n,t}$, where $\mathcal{P}$ can only pass the check with probability at most $3/2^d$, if using some incorrect polynomials $\hat{v}_j(\cdot), j \in [t]$. Note that, unlike in the basic protocol, we do not need re-embedding pair MAC in this step (the PAC generation above came with the required guarantee).

The final check is for inputs and outputs. After Λ is opened to $\mathcal{P}$, the two parties can obtain MACs $[u_j(\Lambda)]$, $j \in [n]$. Note that they have already obtained MACs for inputs. 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 PACs for inputs are consistent with inputs. Checking output can be done by opening the PAC associated with the output wire, since a certain equation will hold with honest execution of the protocol.

The protocol is given in Figure 6,7, which takes the `BatchCheck` procedure in Figure 5 and Π_{PAC} in Figure 21 as sub-protocols. We have the following theorem that guarantees security. We provide a sketched proof in Section C.3 in Supplementary Material.

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

The protocol $\Pi_{\text{slZK}}^{m,n,t}$ communicates 2 AHE ciphertexts per multiplication gate. The main costs for checking multiplications consist of two parts: transferring random coefficients $\boldsymbol{\chi}$ in Step 5.*e* in Figure 7, 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.

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 a SIMD circuit (cf. [19,12]) 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 [42] is specific to proving circuit satisfiability. Since

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-d_1 polynomial and $g_i(\cdot)$ is a degree-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 $i \in [m], j \in [l]$ as follows.

Linear combination phase: Before the polynomial key Λ is opened, $\mathcal{P}$ and $\mathcal{V}$ do as follows.

1. $\mathcal{P}$ 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 $\mathtt{Pre} - \mathtt{Gen}$ procedure of Π_{PAC} with input 2, to pre-generate two PACs $[r(\cdot)]$ and $[s(\cdot)]$.
2. $\mathcal{V}$ samples a $\mathtt{seed} \leftarrow \{0,1\}^\kappa$ and sends it to $\mathcal{P}$. Then, two parties computes $(\chi_1, ..., \chi_l) := H(\mathtt{seed}) \in \mathbb{Z}_{2^k}^l$.
3. $\mathcal{P}$ 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.

Check phase:

1. $\mathcal{P}$ and $\mathcal{V}$ locally compute $[\mu] := [f(\Lambda)] - f(\Lambda)$ and $[\nu] := [g(\Lambda)] - g(\Lambda)$. Then, $\mathcal{P}$ sends M_μ, M_ν to $\mathcal{V}$, and $\mathcal{V}$ checks $M_\mu = K_\mu$ and $M_\nu = K_\nu$, i.e. $\mathcal{P}$ opens $[\mu]$ and $[\nu]$ to $\mathcal{V}$. If the check fails, $\mathcal{V}$ aborts.

Fig. 5: Procedure for checking the τ-consistency of polynomial evaluations for two sets of PACs.

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

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 $m = m_1 m_2$ witnesses $\boldsymbol{w}_i \in \mathbb{Z}_{2^k}^n$ such that $\mathcal{C}(\boldsymbol{w}_i) = 1$, $i \in [m]$. Let (ϕ, ψ) be a RMFE pair, and $\zeta \in \mathtt{GR}(2^k, d)$ is of order $2^d - 1$. Let T be the set $\{0, 1, \zeta, ..., \zeta^{2^d-2}\}$, and $\alpha_1, ..., \alpha_{m_2}$ be distinct elements of T. Let $\xi_i(X) = \prod_{j\in[m_2], j\neq i}(X - \alpha_j)/(\alpha_i - \alpha_j) \in \mathtt{GR}(2^k, d)[X]$ be a degree-$(m_2 - 1)$ polynomial for each $i \in [m_2]$, which is referred to as a Lagrange polynomial.

Offline phase The offline phase is independent of $\mathcal{C}$ and just needs upper bounds on the number of inputs and multiplication gates of $\mathcal{C}$ as input.

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 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 μ_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 $W^i = (\boldsymbol{w}_{(i-1)\cdot m_1+1}, \boldsymbol{w}_{(i-1)\cdot m_1+2}, ..., \boldsymbol{w}_{(i-1)\cdot m_1+m_1})$, $\mathcal{P}$ computes $\boldsymbol{\omega^i} := \phi(W^i)$, 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-$(m_2 - 1)$ 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-$(2m_2 - 2)$ 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-$(m_2 - 1)$ 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 $[\widetilde{u}_c(\cdot)]$.

5. $\mathcal{P}$ and $\mathcal{V}$ do 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 $(a, b, c, \mathtt{Mul}) \in \mathcal{C}$, $\mathcal{P}$ and $\mathcal{V}$ now hold $[u_a(\Lambda)], [u_b(\Lambda)], [\widetilde{u}_c(\Lambda)]$. $\mathcal{P}$ computes $A_{0,j} := M_{u_a(\Lambda)} \cdot M_{u_b(\Lambda)} \in \mathtt{GR}(2^k, d)$ and $A_{0,j} := u_a(\Lambda) \cdot M_{u_b(\Lambda)} + u_a(\Lambda) \cdot M_{u_a(\Lambda)} - M_{\widetilde{u}_c(\Lambda)} \in \mathtt{GR}(2^k, d)$. $\mathcal{V}$ computes $B_j := K_{u_a(\Lambda)} \cdot K_{u_b(\Lambda)} - \Delta \cdot K_{\widetilde{u}_c(\Lambda)} \in \mathtt{GR}(2^k, d)$.

Fig. 6: 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

(c) $\mathcal{P}$ and $\mathcal{V}$ execute the check phase of the `BatchCheck` procedure to complete the above check. If the check fails, $\mathcal{V}$ aborts.

(d) $\mathcal{P}$ sets $A_0^* := M_\pi, A_1^* := \pi$, and $\mathcal{V}$ sets $B^* := K_\pi$ so that $B^* = A_0^* + \Delta \cdot A_1^*$.

(e) $\mathcal{V}$ draws a uniformly random $\boldsymbol{\chi}$ from $\mathbb{F}_{2^d}^t$ and sends it to $\mathcal{P}$.

(f) $\mathcal{P}$ computes $X := \sum_{j\in[t]} \chi_j \cdot A_{0,j} + A_0^* \in \mathtt{GR}(2^k, d)$ and $Y := \sum_{j\in[t]} \chi_j \cdot A_{1,j} + A_1^* \in \mathtt{GR}(2^k, d)$, and sends (X, Y) to $\mathcal{V}$.

(g) $\mathcal{V}$ computes $Z := \sum_{j\in[t]} \chi_j \cdot B_j + B^* \in \mathtt{GR}(2^k, d)$, and checks whether $Z = X + \Delta \cdot Y$ holds. If the check fails, $\mathcal{V}$ outputs `false` and aborts.

6. $\mathcal{P}$ and $\mathcal{V}$ do the following input 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.

(a) For each $j \in [n]$, $\mathcal{P}$ and $\mathcal{V}$ locally compute an MAC $[z_j] := \sum_{i\in[m_2]} \xi_i(\Lambda) \cdot [\omega_j^i] - [u_j(\Lambda)]$. Then, $\mathcal{P}$ opens $[z_j]$, $\mathcal{V}$ continues if and only if $z_j = 0$ holds for all $j \in [n]$. Otherwise, $\mathcal{V}$ aborts.

(b) Let $[u(\cdot)]$ be the PAC associated with the output wire of $\mathcal{C}$. $\mathcal{P}$ sends $M_{u(\Lambda)}$ to $\mathcal{V}$.

(c) $\mathcal{V}$ checks whether $K_{u(\Lambda)} = M_{u(\Lambda)} + \Delta \cdot \phi(\mathbf{1})$. If the check fails, $\mathcal{V}$ outputs `false`. Otherwise, $\mathcal{V}$ outputs `true`.

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 II**.

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 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 sublinear communication variant, 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 (now it allows to check the τ-consistency, namely all $\mathbb{Z}_{2^k}$-components). It is also possible to combine the above two methods, where we view the SIMD circuit over $\mathbb{Z}_{2^k}$ arranged by the first method as a generic circuit over $\mathtt{GR}(2^k, d)$ and use the second method to deal with it. Users are advised to pick the most efficient strategy according to the circuit topology.

4 Reusable VOLE over Galois Rings

In this section, we show how to distribute VOLE correlations over Galois rings with sublinear communication complexity. We begin by a novel base VOLE con-

struction over Galois rings. Then we present two PCG-style VOLE constructions, where the first basic construction is more efficient (but has multiple rounds), and the second is a two-round variant we need for building a MDV-NIZK protocol.

4.1 Reusable VOLE over Galois rings

Our idea comes from the fact that every Galois ring contains a Galois field and computations over Galois rings can be emulated by computations over Galois fields. More specifically, any $a \in \mathtt{GR}(p^k, d)$ can be uniquely written as $a_0 + a_1 \cdot p + ... + a_{k-1} \cdot p^{k-1}$, where $a_i \in \mathbb{F}_{p^d}$. For any $a, b \in \mathtt{GR}(p^k, d)$, $a + b$ can be computed as

$$a + b = (a_0 + b_0) + (a_1 + b_1) \cdot p + ... + (a_{k-1} + b_{k-1}) \cdot p^{k-1},$$

and $a \cdot b$ can be computed as

$$\begin{aligned} a \cdot b &= a_0 b_0 + (a_0 b_1 + a_1 b_0) \cdot p + ... + (a_0 b_{k-1} + ... + a_{k-1} b_0) \cdot p^{k-1} \\ &= \sum_{m=0}^{k-1} \sum_{i+j=m} a_i b_j \cdot p^m. \end{aligned}$$

Thus, we utilize the above algebraic structures of Galois rings to realize a semi-honest secure VOLE protocol over $\mathtt{GR}(2^k, d)$ in the $\mathcal{F}_{\mathrm{vOLE}}^{\mathbb{F}_{2^d}}$-hybrid model in Figure 8, where $\mathcal{F}_{\mathrm{vOLE}}^{\mathbb{F}_{2^d}}$ (see Figure 15) is a fixed input VOLE functionality. We remark that if building VOLE over $\mathbb{F}_{2^d}$ on OTs, our construction reduces the communication complexity by half approximately comparing to that constructing VOLE over $\mathtt{GR}(2^k, d)$ directly from OTs in the semi-honest setting.

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

$P_{\mathcal{R}}$ has input $\Delta \in \mathtt{GR}(2^k, d)$, $P_{\mathcal{S}}$ has input $\boldsymbol{a}, \boldsymbol{b} \in \mathtt{GR}(2^k, d)^l$, where l is a length parameter. We write $\Delta := \Delta_0 + \Delta_1 \cdot 2 + ... + \Delta_{k-1} \cdot 2^{k-1}$, where $\Delta_i \in \mathbb{F}_{2^d}$ for $i = 0, 1, ..., k-1$. Similarly, we define $\boldsymbol{a}_i, \boldsymbol{b}_i$ for $i = 0, 1, ..., k-1$.

(1) For $i = 0, 1, ..., k-1$, $P_{\mathcal{R}}$ sends $(i; \Delta_i)$ to $\mathcal{F}_{\mathrm{vOLE}}^{\mathbb{F}_{2^d}}$.
(2) For $i = 0, 1, ..., k-1$, $P_{\mathcal{S}}$ picks random $\boldsymbol{b}_{s,t} \in \mathbb{F}_{2^d}^l$ such that $\sum_{s,t}^{s+t=i} \boldsymbol{b}_{s,t} = \boldsymbol{b}_i$.
(3) For $i = 0, 1, ..., k-1$, and $j = 0, ..., k-1-i$, $P_{\mathcal{S}}$ sends $(i; \boldsymbol{a}_j, \boldsymbol{b}_{i,j}; j)$ to $\mathcal{F}_{\mathrm{vOLE}}^{\mathbb{F}_{2^d}}$.
(4) Upon receiving $(i; \boldsymbol{v}_{i,j}; j)$ where $\boldsymbol{v}_{i,j} = \boldsymbol{a}_j \cdot \Delta + \boldsymbol{b}_{i,j}$, for $i = 0, 1, ..., k-1$, and $j = 0, , ..., k-1-i$, $P_{\mathcal{R}}$ computes $\boldsymbol{v} := \boldsymbol{v}_{0,0} + (\boldsymbol{v}_{0,1} + \boldsymbol{v}_{1,0}) \cdot 2 + ... + (\boldsymbol{v}_{0,k-1} + ... + \boldsymbol{v}_{k-1,0}) \cdot 2^{k-1}$.

Fig. 8: Protocol for VOLE over $\mathtt{GR}(2^k, d)$ in the $\mathcal{F}_{\mathrm{vOLE}}^{\mathbb{F}_{2^d}}$-hybrid model.

We have the following theorem.

Theorem 4. *Protocol $\Pi_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$ perfectly realizes the functionality $\mathcal{F}_{\mathrm{vOLE}}^{\mathtt{GR}(2^k,d)}$ with semi-honest security in the $\mathcal{F}_{\mathrm{vOLE}}^{\mathbb{F}_{2^d}}$-hybrid model. The total length of the VOLE instances over $\mathbb{F}_{2^d}$ is $\frac{k(k+1)}{2} \cdot l$.*

Proof. If $P_{\mathcal{R}}$ is corrupted. The simulator $S_{\mathcal{R}}$ receives $\Delta_0, ..., \Delta_{k-1} \in \mathbb{F}_{2^d}$ from the adversary, and computes $\Delta := \sum_{i=0}^{k-1} \Delta_i \cdot 2^i$. Then $S_{\mathcal{R}}$ sends Δ to $\mathcal{F}_{\mathrm{vOLE}}^{\mathtt{GR}(2^k,d)}$, and receives $\boldsymbol{v}$ from $\mathcal{F}_{\mathrm{vOLE}}^{\mathtt{GR}(2^k,d)}$. $S_{\mathcal{R}}$ writes $\boldsymbol{v}$ as $\sum_{i=0}^{k-1} \boldsymbol{v}_i \cdot 2^i$. $S_{\mathcal{R}}$ sets $\boldsymbol{v}_{0,0} := \boldsymbol{v}_0$ and samples $\boldsymbol{v}_{j,i-j} \xleftarrow{\$} \mathbb{F}_{2^d}^l$ such that $\sum_{j=0}^{i} \boldsymbol{v}_{j,i-j} = \boldsymbol{v}_i$, for $j = 0, ..., i$ and $i = 1, ..., k-1$. Finally, $S_{\mathcal{R}}$ sends these $\boldsymbol{v}_{i,j}$ to the adversary. The indistinguishability is clear since these $\boldsymbol{v}_{i,j}$ are identically distributed both in the real world and the ideal world.

If $P_{\mathcal{S}}$ is corrupted. The simulator $S_{\mathcal{S}}$ receives $\boldsymbol{a}_j, \boldsymbol{b}_{i,j}$ from the adversary for $i = 0, ..., k-1$ and $j = 0, ..., k-1-i$. $S_{\mathcal{S}}$ computes $\boldsymbol{a} := \sum_{i=0}^{k-1} \boldsymbol{a}_i \cdot 2^i$ and $\boldsymbol{b} := \sum_{i=0}^{k-1} (\sum_{j=0}^{i} \boldsymbol{b}_{j,i-j}) \cdot 2^i$. $S_{\mathcal{S}}$ sends $\boldsymbol{a}, \boldsymbol{b}$ to $\mathcal{F}_{\mathrm{vOLE}}^{\mathtt{GR}(2^k,d)}$. The indistinguishability is clear since the outputs in the real world and the ideal world are identical.

For reusable security against a malicious sender, where $\mathcal{A}$ can cheat by taking inconsistent inputs to VOLE instances over $\mathbb{F}_{2^d}$, we use the eVOLE technique in [22], while there is no room for a malicious receiver to cheat. Thus, instantiating $\mathcal{F}_{\mathrm{vOLE}}^{\mathbb{F}_{2^d}}$ with the Paillier-based reusable VOLE construction in [14], we obtain a reusable VOLE over $\mathtt{GR}(2^k, d)$ in the fully malicious setting. We have the following theorem.

Theorem 5. *There exists a malicious secure reusable VOLE construction over $\mathtt{GR}(2^k, d)$ in the $\mathcal{F}_{\mathrm{vOLE}}^{\mathbb{F}_{2^d}}$-hybrid model, where the receiver's input Δ is uniformly random. The total length of the VOLE instances over $\mathbb{F}_{2^d}$ is $\frac{(3k-1)k}{2} \cdot l$, and $P_{\mathcal{S}}$ additionally sends $\frac{k(k-1)}{2} \cdot l$ elements over $\mathbb{F}_{2^d}$ to $P_{\mathcal{R}}$.*

Proof. Correctness and security are immediate from the correctness and the security of the underlying protocols, where the receiver's input Δ is uniformly random as eVOLE requires. As illustrated in [22], eVOLE requires two additional VOLE correlations and transferring one field element for proving one equality constraint. Since there are $\frac{k(k-1)}{2} \cdot l$ equality constraints need to prove, the total length of the VOLE instances over $\mathbb{F}_{2^d}$ is $\frac{k(k+1)}{2} \cdot l + k(k-1) \cdot l = \frac{(3k-1)k}{2} \cdot l$, and $\frac{k(k-1)}{2} \cdot l$ elements over $\mathbb{F}_{2^d}$ are sent from $P_{\mathcal{S}}$ to $P_{\mathcal{R}}$.

4.2 Basic VOLE Construction

Following the established VOLE extension paradigm, which extends short base VOLE correlations into a large number of VOLE correlations via LPN assumptions, we instantiate the $(\mathcal{D}, \mathcal{G}, \mathcal{R})$-LPN assumption (we refer to Section B in Supplementary Material for LPN preliminaries) with $\mathcal{R}$ a Galois ring $\mathtt{GR}(2^k, d)$, $\mathcal{D}$ the fixed Hamming weight distribution and $\mathcal{G}$ the random linear codes generation algorithm. The above LPN instantiation also covers the case of regular noise as

shown in [35], on which the discussion about existing attacks can be found in Section B.3. To achieve better efficiency, we use the regular Hamming weight distribution, where the error vector of weight t can be easily divided into t blocks with each block having only one non-zero entry. To our best knowledge, no attack has been found to utilize the regular structure and the ring structure to decrease the cost except for a simple reduction attack (we refer to Section B.2). For example, one can reduce an LPN instance's noise weight by approximately $1/2^d$ fraction via modulo 2, since $\mathtt{GR}(2^k, d)/(2) \cong \mathbb{F}_{2^d}$, which may be tolerable if d is sufficiently large. We see this attack is much less efficient than that for LPN over $\mathbb{Z}_{2^k}$. Alternatively, similar to [3,35], one can additionally require that each entry of the error vector $\boldsymbol{e}$ is invertible.

We use the GGM tree [28] based puncturable pseudorandom functions [7] (PPRF for short) construction to obtain VOLE correlations for such error vectors. Informally in a PPRF, one party P_1 generates a key s and the other party P_2 randomly picks an index $\alpha \in [0, n)$, after engaging in a protocol, P_2 obtains a punctured key $s\{\alpha\}$. P_1 can use the key s to obtain n pseudorandom values, while P_2 can use the punctured key $s\{\alpha\}$ to obtain these values except one value that is dependent on the choice of α. The GGM algorithms are presented in Figure 20.

Clearly, one can verify that on inputs $\alpha, \{K_{\bar{\alpha}_i, i}\}_{i\in[h]}$, the outputs of $\mathtt{GGM.Eval}$ are consistent to those computed by $\mathtt{GGM.Gen}$. Hence, taking s as the key of PPRF, $\{K_{\bar{\alpha}_i, i}\}_{i\in[h]}$ is the punctured key for index α. Notice that $\mathtt{GGM.Gen}$ also outputs $\{\Gamma_j\}_{j\in[0,2^{h-1})}$, which can be used for checking consistency of the GGM tree in our two-round variant and it will be explained later.

Our basic VOLE construction takes single-point VOLE as a black-box building block, which is a variant of VOLE, where the sender $P_{\mathcal{S}}$ holds $\boldsymbol{w}, \boldsymbol{u} \in \mathtt{GR}(2^k, d)^n$ with $\boldsymbol{u}$ having only one entry invertible in $\mathtt{GR}(2^k, d)$ and zeros everywhere else, and the receiver $P_{\mathcal{R}}$ holds $\boldsymbol{v} \in \mathtt{GR}(2^k, d)^n, \Delta \in \mathtt{GR}(2^k, d)$ such that $\boldsymbol{w} = \boldsymbol{v} - \Delta \cdot \boldsymbol{u}$. We formalize the ideal functionality for single-point VOLE in Figure 17.

As the GGM tree based PPRFs allow two parties $P_{\mathcal{S}}$ and $P_{\mathcal{R}}$ to obtain $\{v_j\}_{j\in[0,n)\setminus\{\alpha\}}$ and $\{v_j\}_{j\in[0,n)}$, when they are given the punctured key corresponding to α and the PPRF key, respectively. Delivering the punctured key can be done by invoking the functionality for oblivious transfer. More specifically, $P_{\mathcal{R}}$ sends $K_{0,i}, K_{1,i}$ to $\mathcal{F}_{\mathrm{OT}}$ and $P_{\mathcal{S}}$ sends $\bar{\alpha}_i$ to $\mathcal{F}_{\mathrm{OT}}$, for $i \in [h]$, then $P_{\mathcal{S}}$ will receive the punctured key.

After receiving the punctured key, $P_{\mathcal{S}}$ can set $u_j := 0$, and $w_j := v_j = v_j - \Delta \cdot u_j$, for $j \neq \alpha$. To obtain the VOLE correlation on position α, they first obtain a VOLE correlation $\delta = \gamma - \Delta \cdot \beta$, where β is invertible in $\mathtt{GR}(2^k, d)$. Then, by sending $g := \gamma - \sum_{j\in[0,n)} v_j$ to $P_{\mathcal{S}}$, $P_{\mathcal{S}}$ can compute $w_\alpha := \delta - (g + \sum_{i\neq\alpha} w_i)$, such that $w_\alpha = v_\alpha - \Delta \cdot \beta$ holds. Letting $u_\alpha := \beta$, $P_{\mathcal{S}}$ and $P_{\mathcal{R}}$ obtain a single-point VOLE correlation $\boldsymbol{w} = \boldsymbol{v} - \Delta \cdot \boldsymbol{u}$.

To guarantee the honest behaviors, we use a random linear combination check on the single-point VOLE correlation. 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

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

Obviously, this equation can not be directly used for consistency check, since $\Delta \cdot \chi_\alpha \cdot \beta$ can not be computed by only one party and the linear combination will leak some information. These can be solved by masking the linear combination equation with a VOLE correlation where $z = y - \Delta \cdot \chi_\alpha \cdot \beta$. Namely, the two parties check the equation

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

We remark that this check provides a selective failure attack for the corrupted receiver, where $\mathcal{A}$ can guess a set $I \subseteq [0,n)$ that contains α and provides inconsistent inputs to $\mathcal{F}_{\mathrm{OT}}$ according to I. Besides, it also enables the corrupted sender to cheat on generating the VOLE correlation for $\chi_\alpha \cdot \beta$ and $\mathcal{A}$ will learn Δ modulo some 2^s if the attack succeeds. Thus, we let the functionality $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$ allow these two kinds of leakage.

We give the single-point VOLE protocol in Figure 9, and we have the following theorem that guarantees security. The proof can be found in Section D.1 in Supplementary Material.

Theorem 6. *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 $1/2^d + \mathsf{negl}(\kappa)$.*

We build a VOLE extension protocol that relies on the single-point VOLE and a variant of the LPN assumption. Let $A \in \mathtt{GR}(2^k,d)^{m\times n}$ be a generating matrix and $\mathcal{D}$ be a noise distribution over $\mathtt{GR}(2^k,d)^n$, for which the primal LPN assumption holds. Suppose the two parties have obtained $m+n$ VOLE correlations $\boldsymbol{w} = \boldsymbol{v} - \Delta \cdot \boldsymbol{u}$ and $\boldsymbol{c} = \boldsymbol{b} - \Delta \cdot \boldsymbol{e}$, where $\boldsymbol{u} \in \mathtt{GR}(2^k,d)^m$ and $\boldsymbol{e} \leftarrow \mathcal{D}$. They can locally obtain n random VOLE correlations over $\mathtt{GR}(2^k,d)$, since we have that

$$\boldsymbol{w} \cdot A + \boldsymbol{c} = (\boldsymbol{v} \cdot A + \boldsymbol{b}) - \Delta \cdot (\boldsymbol{u} \cdot A + \boldsymbol{e}),$$

and $\boldsymbol{u} \cdot A + \boldsymbol{e}$ is pseudorandom by the primal LPN assumption. For simplicity, we assume the primal LPN assumption with the regular noise distribution, and we can obtain VOLE correlations for $\boldsymbol{e}$ by executing multiple $\Pi_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$ in parallel and concatenating the outputs. In addition, as showed in [41], a kind of "bootstrapping" technique can be used to reduce communication complexity, where a part of the outputs can be reserved as the base VOLE correlations for the next extension. We give the VOLE extension protocol in Figure 10.

Notice that our single-point VOLE protocol reveals some information on the non-zero entry to the adversaries, thus the LPN assumption should be defined

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

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

SP-Extend: On input $n = 2^h$, the two parties 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 $c = b - \Delta \cdot a$ holds.

(b) $P_{\mathcal{S}}$ samples $\alpha \xleftarrow{\$} [0, n)$, and draws a uniformly random β in the set of units of $\mathtt{GR}(2^k, d)$. $P_{\mathcal{S}}$ sets the vector $\boldsymbol{u} \in \mathtt{GR}(2^k, d)^n$ with $u_\alpha = \beta$ and $u_i = 0$ for all $i \neq \alpha$.

(c) $P_{\mathcal{S}}$ sets $\delta := c$ and sends $a' := \beta - a \in \mathtt{GR}(2^k, d)$ to $P_{\mathcal{R}}$. $P_{\mathcal{R}}$ computes $\gamma := b + \Delta \cdot a' \in \mathtt{GR}(2^k, d)$. Note that $\delta = \gamma - \Delta \cdot \beta$, 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 $(\{v_j\}_{j\in[0,n)}, \{(K_0^i, K_1^i)\}_{i\in[h]})$. 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 $\{v_j\}_{j\neq\alpha}$.

(e) $P_{\mathcal{R}}$ sends $g := \gamma - \sum_{j\in[0,n)} v_j \in \mathtt{GR}(2^k, d)$ to $P_{\mathcal{S}}$. $P_{\mathcal{S}}$ defines a vector $\boldsymbol{w} \in \mathtt{GR}(2^k, d)^n$ such that $w_i := v_i$ for $i \neq \alpha$ and $w_\alpha := \delta - (g + \sum_{i\neq\alpha} w_i)$. Note that $\boldsymbol{w} = \boldsymbol{v} - \Delta \cdot \boldsymbol{u}$.

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 $z = y^* - \Delta \cdot x$ holds.

(b) $P_{\mathcal{S}}$ samples $\chi_i \xleftarrow{\$} \mathtt{GR}(2^k, d)$ for $i \in [0, n)$, and computes $x^* := \chi_\alpha \cdot \beta - x \in \mathtt{GR}(2^k, d)$. $P_{\mathcal{S}}$ sends $(\{\chi_i\}_{i\in[0,n)}, x^*)$ to $P_{\mathcal{R}}$. $P_{\mathcal{R}}$ computes $y := y^* + \Delta \cdot x^* \in \mathtt{GR}(2^k, d)$. Now, $z = y - \Delta \cdot \chi_\alpha \cdot \beta$ holds.

(c) $P_{\mathcal{S}}$ computes $V_{P_{\mathcal{S}}} := \sum_{i\in[0,n)} \chi_i \cdot w_i - z$, and sends $V_{P_{\mathcal{S}}}$ to $\mathcal{F}_{\mathrm{EQ}}$ as $\mathcal{P}$. $P_{\mathcal{R}}$ computes $V_{P_{\mathcal{R}}} := \sum_{i\in[0,n)} \chi_i \cdot v_i - y$, and send $V_{P_{\mathcal{R}}}$ to $\mathcal{F}_{\mathrm{EQ}}$ as $\mathcal{V}$. $P_{\mathcal{R}}$ receives $V_{P_{\mathcal{S}}}$. $P_{\mathcal{S}}$ and $P_{\mathcal{R}}$ abort if the equality test fails.

3. **Output:** $P_{\mathcal{S}}$ outputs $(\boldsymbol{w}, \boldsymbol{u})$, and $P_{\mathcal{R}}$ outputs $\boldsymbol{v}$.

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

with some static leakage (see Definition 5). Besides, the VOLE functionality should be redefined with a global key query procedure (see $\mathcal{F}_{\text{qVOLE}}^{\mathtt{GR}(2^k,d)}$ in Figure 16). We have the following theorem, while the proof can be found in Section D.1 in Supplementary Material.

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

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

Init:

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

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

Extend:

(1) For $i \in [t]$, both parties send $(\mathtt{SP\text{-}Extend}, n/t)$ to $\mathcal{F}_{\text{spVOLE}}^{\mathtt{GR}(2^k,d)}$. $P_{\mathcal{S}}$ receives $\boldsymbol{e_i}, \boldsymbol{c_i}$ and $P_{\mathcal{R}}$ receives $\boldsymbol{b_i}$, where $\boldsymbol{c_i} = \boldsymbol{b_i} - \Delta \cdot \boldsymbol{e_i}$ over $\mathtt{GR}(2^k,d)^{n/t}$, and $\boldsymbol{e_i} \in \mathtt{GR}(2^k,d)^{n/t}$ has exactly one entry invertible in $\mathtt{GR}(2^k,d)$ and zeros everywhere else.

(2) Let $\boldsymbol{e} := (\boldsymbol{e_1}, ..., \boldsymbol{e_t}) \in \mathtt{GR}(2^k,d)^n$, $\boldsymbol{c} := (\boldsymbol{c_1}, ..., \boldsymbol{c_t}) \in \mathtt{GR}(2^k,d)^n$ and $\boldsymbol{b} := (\boldsymbol{b_1}, ..., \boldsymbol{b_t}) \in \mathtt{GR}(2^k,d)^n$. $P_{\mathcal{S}}$ computes $\boldsymbol{x} := \boldsymbol{u} \cdot A + \boldsymbol{e} \in \mathtt{GR}(2^k,d)^n$, and $\boldsymbol{M} := \boldsymbol{w} \cdot A + \boldsymbol{c} \in \mathtt{GR}(2^k,d)^n$. $P_{\mathcal{R}}$ computes $\boldsymbol{K} := \boldsymbol{v} \cdot A + \boldsymbol{b} \in \mathtt{GR}(2^k,d)^n$.

(3) $P_{\mathcal{S}}$ updates $\boldsymbol{u}, \boldsymbol{w}$ by setting $\boldsymbol{u} := \boldsymbol{x}[1:m] \in \mathtt{GR}(2^k,d)^m$ and $\boldsymbol{w} := \boldsymbol{M}[1:m] \in \mathtt{GR}(2^k,d)^m$, and outputs $(\boldsymbol{x}[m+1:n], \boldsymbol{M}[m+1:n])$. $P_{\mathcal{S}}$ updates $\boldsymbol{v}$ by setting $\boldsymbol{v} := \boldsymbol{K}[1:m] \in \mathtt{GR}(2^k,d)^m$ and outputs $\boldsymbol{K}[m+1:n]$.

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

Theorem 7. *If the decisional* $(\mathtt{RG}, \mathcal{G}, \mathtt{GR}(2^k,d))$-LPN$(m,n,t)$ *with static leakage assumption holds, then* $\Pi_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$ *UC-realizes* $\mathcal{F}_{\text{qVOLE}}^{\mathtt{GR}(2^k,d)}$ *in the* $\mathcal{F}_{\text{spVOLE}}^{\mathtt{GR}(2^k,d)}$*-hybrid model.*

4.3 Two-Round Variant

Here we focus on reducing the round complexity of the VOLE extension protocol. Our basic construction can not be made two-round, since the two parties need to obtain an additional VOLE correlation for the check. Inspired by the approach of [8], we construct a two-round VOLE extension protocol that relies on the ideal functionality for generalized VOLE over Galois rings (see $\mathcal{F}_{\text{gVOLE}}^{\mathtt{GR}(2^k,d)}$ in Figure 18). Informally in the generalized VOLE, the two parties obtain two VOLE correlations with the same length for different global keys, e.g. $\boldsymbol{M} = \boldsymbol{K} - \Delta \cdot \boldsymbol{y}$ and $\boldsymbol{c} = \boldsymbol{a} - \delta \cdot \boldsymbol{y}$. Similar to $\Pi_{\text{rVOLE}}^{\mathtt{GR}(2^k,d)}$, the functionality $\mathcal{F}_{\text{gVOLE}}^{\mathtt{GR}(2^k,d)}$ can be realized

by VOLE correlations over $\mathbb{F}_{2^d}$, where the eVOLE technique [22] can be applied as well.

We provide a multi-point VOLE construction in Figure 11, which can be viewed as a concatenation of multiple single-point VOLEs intuitively. The ideal functionality for multi-point VOLE is presented in Figure 19. $P_{\mathcal{S}}$ and $P_{\mathcal{R}}$ can naturally obtain pseudorandom VOLE correlations with the help of 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, whose formal definition is given in Definition 6.

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 slightly modify the $\mathtt{GGM.Eval}$ algorithm and obtain $\mathtt{GGM.Eval'}$ in Figure 20. To ensure all the right leaves fix a unique tree, we require that G' has the right half injective property [3].

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

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

We have the following theorem, whose proof can be found in Section D.2 in Supplementary Material.

Theorem 8. *If G and G' are PRGs with G' having the right half injective property, and $\mathtt{Hash} : \{0,1\}^{2^h\kappa} \mapsto \{0,1\}^\kappa$ is a collision resistant hash function, $\Pi_{\mathrm{mpVOLE}}^{\mathtt{GR}(2^k,d)}$ realizes the functionality $\mathcal{F}_{\mathrm{mpVOLE}}^{\mathtt{GR}(2^k,d)}$ in the $(\mathcal{F}_{\mathrm{gVOLE}}^{\mathtt{GR}(2^k,d)}, \mathcal{F}_{\mathrm{OT}})$-hybrid model with malicious security.*

By using 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 correlations is small, or namely the "bootstrapping" in the basic construction is not activated, the two-round variant has smaller communication complexity, as dual-LPN allows polynomial stretch. Instantiating base VOLEs with the reusable construction and together with the basic ZK protocol, we obtain a reusable MDV-NIZK in the ROM.

[3] As noted in [8], 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.

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

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

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

Construct:

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

(b) For $j \in [t]$, both parties do as follows. $P_{\mathcal{R}}$ computes $s^j \leftarrow \mathtt{GGM.KeyGen}(1^\kappa)$, and runs $\mathtt{GGM.Gen}(2^{h+1}, s^j)$ to obtain $(\{(v^j_{2i}, v^j_{2i+1}, \Gamma^j_i)\}_{i\in[0,2^h)}, \{(K^j_{0,i}, K^j_{1,i})\}_{i\in[h+1]})$. Recompute $K^j_{1,h+1} := \bigoplus_{i\in[0,2^h)} \Gamma^j_i$. Let $\alpha^j = \sum_{i\in[h]} 2^{h-i}\alpha^j_i$, where $\alpha^j_i \xleftarrow{\$} \{0,1\}$, and let $\bar{\alpha}^j_i$ denote the complement of α^j_i. For $i \in [h]$, $P_{\mathcal{S}}$ sends $\bar{\alpha}^j_i$ to $\mathcal{F}_{\mathrm{OT}}$ while $P_{\mathcal{R}}$ sends $(K^j_{0,i}, K^j_{1,i})$ to $\mathcal{F}_{\mathrm{OT}}$, then $P_{\mathcal{S}}$ receives $K^j_{\bar{\alpha}^j_i,i}$. $P_{\mathcal{R}}$ sends $K^j_{1,h+1}$ to $P_{\mathcal{S}}$. $P_{\mathcal{S}}$ runs $\mathtt{GGM.Eval'}(\alpha^j, \{K^j_{\bar{\alpha}^j_i,i}\}_{i\in[h]}, K^j_{1,h+1})$ and gets $(\{(v^j_{2i}, v^j_{2i+1})\}_{i\neq\alpha^j}, \{\Gamma^j_i\}_{i\in[0,2^h)})$. Additionally, $P_{\mathcal{R}}$ sends $g^j_0 := K_j - \sum_{i\in[0,2^h)} v^j_{2i}$ and $g^j_1 := a_j - \sum_{i\in[0,2^h)} v^j_{2i+1} \in \mathtt{GR}(2^k,d)$ to $P_{\mathcal{S}}$.

(c) $P_{\mathcal{S}}$ defines vectors $\boldsymbol{w}^j_0, \boldsymbol{w}^j_1 \in \mathtt{GR}(2^k,d)^{2^h}$ such that $w^j_{0,i} = v^j_{2i}$, $w^j_{1,i} = v^j_{2i+1}$ for $i \neq \alpha^j$ and $w^j_{0,\alpha^j} := M_j - (g^j_0 + \sum_{i\neq\alpha^j} w^j_{0,i})$, $w^j_{1,\alpha^j} := c_j - (g^j_1 + \sum_{i\neq\alpha^j} w^j_{1,i})$. $P_{\mathcal{R}}$ defines vectors $\boldsymbol{v}^j_0 := (v^j_0, v^j_2, ..., v^j_{2^{h+1}-2})$ and $\boldsymbol{v}^j_1 := (v^j_1, v^j_3, ..., v^j_{2^{h+1}-1})$. Note that $\boldsymbol{w}^j_0 = \boldsymbol{v}^j_0 - \Delta \cdot \boldsymbol{e}^j$ and $\boldsymbol{w}^j_1 = \boldsymbol{v}^j_1 - \delta \cdot \boldsymbol{e}^j$, where $\boldsymbol{e}^j \in \mathtt{GR}(2^k,d)^{2^h}$ has only one entry invertible in $\mathtt{GR}(2^k,d)$ and zeros everywhere else, i.e. $e^j_{\alpha^j} = y_j \in \mathtt{GR}(2^k,d)^*$ and $e^j_i = 0$ for $i \neq \alpha^j$.

(c) $P_{\mathcal{R}}$ computes $\Gamma^j := \mathtt{Hash}(\Gamma^j_0, ..., \Gamma^j_{2^h-1})$ and sends it to $P_{\mathcal{S}}$. $P_{\mathcal{S}}$ computes $\hat{\Gamma}^j := \mathtt{Hash}(\Gamma^j_0, ..., \Gamma^j_{2^h-1})$ and checks $\Gamma^j = \hat{\Gamma}^j$, for all $j \in [t]$. If the check fails, $P_{\mathcal{S}}$ aborts.

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

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

Output: $P_{\mathcal{S}}$ outputs $(\boldsymbol{w}^1_0, ..., \boldsymbol{w}^t_0) \in \mathtt{GR}(2^k,d)^n, (\boldsymbol{e}^1, ..., \boldsymbol{e}^t) \in \mathtt{GR}(2^k,d)^n$. $P_{\mathcal{R}}$ outputs $(\boldsymbol{v}^1_0, ..., \boldsymbol{v}^t_0) \in \mathtt{GR}(2^k,d)^n$.

Fig. 11: Protocol for multi-point VOLE over $\mathtt{GR}(2^k,d)$ in the $(\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{gVOLE}}, \mathcal{F}_{\mathrm{OT}})$-hybrid model.

References

1. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: Lightweight sublinear arguments without a trusted setup. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 2087–2104 (2017)
2. 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: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 192–211 (2021)
3. 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: Annual International Cryptology Conference. pp. 329–358. Springer (2022)
4. Baum, C., Jadoul, R., Orsini, E., Scholl, P., Smart, N.P.: Feta: Efficient threshold designated-verifier zero-knowledge proofs. Cryptology ePrint Archive (2022), `https://eprint.iacr.org/2022/082`
5. 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: Annual International Cryptology Conference. pp. 92–122. Springer (2021)
6. Blum, A., Furst, M., Kearns, M., Lipton, R.J.: Cryptographic primitives based on hard learning problems. In: Annual International Cryptology Conference. pp. 278–291. Springer (1993)
7. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector OLE. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 896–912 (2018)
8. 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: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 291–308 (2019)
9. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudorandom correlation generators: Silent OT extension and more. In: Annual International Cryptology Conference. pp. 489–518. Springer (2019)
10. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Proceedings 42nd IEEE Symposium on Foundations of Computer Science. pp. 136–145. IEEE (2001)
11. Cascudo, I., Cramer, R., Xing, C., Yuan, C.: Amortized complexity of information-theoretically secure MPC revisited. In: Annual International Cryptology Conference. pp. 395–426. Springer (2018)
12. Cascudo, I., Gundersen, J.S.: A secret-sharing based MPC protocol for boolean circuits with good amortized complexity. In: Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part II. vol. 12551, pp. 652–682. Springer (2020)
13. 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: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1825–1842. ACM (2017)
14. Chase, M., Dodis, Y., Ishai, Y., Kraschewski, D., Liu, T., Ostrovsky, R., Vaikuntanathan, V.: Reusable non-interactive secure computation. In: Annual International Cryptology Conference. pp. 462–488. Springer (2019)

15. 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: Annual International Cryptology Conference. pp. 426–456. Springer (2021)
16. Couteau, G., Rindal, P., Raghuraman, S.: Silver: Silent vole and oblivious transfer from hardness of decoding structured ldpc codes. In: Annual International Cryptology Conference. pp. 502–534. Springer (2021)
17. Cramer, R., Damgård, I., Escudero, D., Scholl, P., Xing, C.: Spd$\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for dishonest majority. In: Annual International Cryptology Conference. vol. 10992, pp. 769–798. Springer (2018)
18. 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: Annual International Cryptology Conference. vol. 12827, pp. 656–686. Springer (2021)
19. Damgård, I., Zakarias, S.: Constant-overhead secure computation of boolean circuits using preprocessing. In: Theory of Cryptography Conference. pp. 621–641. Springer (2013)
20. Debris-Alazard, T., Tillich, J.: Statistical decoding. In: 2017 IEEE International Symposium on Information Theory (ISIT). pp. 1798–1802. IEEE (2017)
21. Dittmer, S., Ishai, Y., Lu, S., Ostrovsky, R.: Improving line-point zero knowledge: Two multiplications for the price of one. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. pp. 829–841 (2022)
22. Dittmer, S., Ishai, Y., Ostrovsky, R.: Line-point zero knowledge and its applications. In: 2nd Conference on Information-Theoretic Cryptography, ITC 2021, July 23-26, 2021, Virtual Conference. LIPIcs, vol. 199, pp. 5:1–5:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021)
23. Escudero, D., Xing, C., Yuan, C.: More efficient dishonest majority secure computation over $\mathbb{Z}_{2^k}$ via galois rings. In: Annual International Cryptology Conference. pp. 383–412. Springer (2022)
24. Esser, A., Kübler, R., May, A.: Lpn decoded. In: Annual International Cryptology Conference. pp. 486–514. Springer (2017)
25. Frederiksen, T.K., Nielsen, J.B., Orlandi, C.: Privacy-free garbled circuits with applications to efficient zero-knowledge. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 191–219. Springer (2015)
26. Ganesh, C., Nitulescu, A., Soria-Vazquez, E.: Rinocchio: snarks for ring arithmetic. Cryptology ePrint Archive (2021)
27. Giacomelli, I., Madsen, J., Orlandi, C.: Zkboo: Faster zero-knowledge for boolean circuits. In: 25th USENIX Security Symposium (USENIX Security 16). pp. 1069–1083 (2016)
28. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM **33**(4), 792–807 (1986)
29. Hazay, C., Venkitasubramaniam, M.: On the power of secure two-party computation. Journal of Cryptology **33**(1), 271–318 (2020)
30. Heath, D., Kolesnikov, V.: Stacked garbling for disjunctive zero-knowledge proofs. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 569–598. Springer (2020)
31. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing. pp. 21–30 (2007)

32. Jawurek, M., Kerschbaum, F., Orlandi, C.: Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In: Proceedings of the 2013 ACM SIGSAC conference on Computer and Communications Security. pp. 955–966 (2013)
33. Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 525–537 (2018)
34. Kim, S., Wu, D.J.: Multi-theorem preprocessing nizks from lattices. Journal of Cryptology **33**(3), 619–702 (2020)
35. Liu, H., Wang, X., Yang, K., Yu, Y.: The hardness of lpn over any integer ring and field for pcg applications. Cryptology ePrint Archive (2022), `https://eprint.iacr.org/2022/712`
36. Lombardi, A., Quach, W., Rothblum, R.D., Wichs, D., Wu, D.J.: New constructions of reusable designated-verifier nizks. In: Annual International Cryptology Conference. pp. 670–700. Springer (2019)
37. Micali, S., Goldreich, O., Wigderson, A.: How to play any mental game. In: Proceedings of the Nineteenth ACM Symp. on Theory of Computing, STOC. pp. 218–229. ACM (1987)
38. Peters, C.: Information-set decoding for linear codes over fq. In: International Workshop on Post-Quantum Cryptography. pp. 81–94. Springer (2010)
39. de Saint Guilhem, C.D., Meyer, L.D., Orsini, E., Smart, N.P.: BBQ: using AES in picnic signatures. In: International Conference on Selected Areas in Cryptography. pp. 669–692. Springer (2019)
40. Wan, Z.X.: Lectures on finite fields and Galois rings. World Scientific Publishing Company (2003)
41. Weng, C., Yang, K., Katz, J., Wang, X.: Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In: 2021 IEEE Symposium on Security and Privacy (SP). pp. 1074–1091. IEEE (2021)
42. Weng, C., Yang, K., Yang, Z., Xie, X., Wang, X.: Antman: Interactive zero-knowledge proofs with sublinear communication. Cryptology ePrint Archive (2022), `https://eprint.iacr.org/2022/566`
43. Yang, K., Sarkar, P., Weng, C., Wang, X.: Quicksilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 2986–3001 (2021)
44. Yang, K., Wang, X.: Non-interactive zero-knowledge proofs to multiple verifiers. Cryptology ePrint Archive (2022), `https://eprint.iacr.org/2022/063`
45. Yang, K., Weng, C., Lan, X., Zhang, J., Wang, X.: Ferret: Fast extension for correlated ot with small communication. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 1607–1626 (2020)
46. Yao, A.C.C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science (sfcs 1986). pp. 162–167. IEEE (1986)
47. Zahur, S., Rosulek, M., Evans, D.: Two halves make a whole - reducing data transfer in garbled circuits using half gates. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 220–250. Springer (2015)

Supplementary Material

A More Preliminaries & Missing Functionalities

Security Model. In UC framework [10], security is defined via the comparison of an *ideal* world and a *real* world. In *real* world, the parties interact with each other following the protocol. In *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 *real* world and *ideal* world, provides the inputs to the parties and can read the outputs. The corrupt party $\mathcal{A}$, controlled by the environment $\mathcal{Z}$, interacts with the honest parties in the *real* world. We consider active adversary and static corruption, namely the adversary $\mathcal{A}$' 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. Thus 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 UC-realize 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 $1/\mathsf{poly}(\kappa)$. For reusable security, we refer to [14,36,22] for a formal definition and a friendly exposition.

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

Additively Homomorphic Encryption. We describe the Additively Homomorphic Encryption (AHE) scheme in the private-key setting, which consists of three algorithms, $\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec}$. The $\mathtt{KeyGen}$ algorithm outputs a secret key sk, which is determined by its random tape. The $\mathtt{Enc}$ algorithm takes the secret key sk and message m as inputs, and outputs a ciphertext $\langle m \rangle$, while the $\mathtt{Dec}$ algorithm decrypts the ciphertext via sk, denoted by $m := \mathtt{Dec}(\langle m \rangle, sk)$. The additive property indicates that the decryption of a linear combination of ciphertexts equals to the corresponding linear combination of plaintexts. We suppose the AHE scheme works on a Galois ring, and satisfies the chosen plaintext attack (CPA) security. For the protocol Π_{PAC} (Figure 21), we require the AHE

scheme additionally satisfy circuit privacy and degree restriction, as shown in [42]. Such AHE schemes exist as shown in [15], and we also show a reduction from AHE schemes over Galois rings to AHE schemes over Galois field later in Section C.3.

Missing Proofs in Preliminaries.

Lemma 5 (Lemma 1, restated). *A nonzero degree-r polynomial over $\mathtt{GR}(p^k, d)$ has at most $rp^{(k-1)d}$ roots.*

Proof. *It can be observed that any element $\alpha \in \mathtt{GR}(p^k, d)$ can be written as $\alpha = \sum_{i\in[0,k)} \alpha_i \cdot p^i$, where $\alpha_i \in \mathbb{F}_{p^d}$. Let $f(x) := \sum_{i\in[0,r]} a_i \cdot x^i$, and $p^t := \gcd(a_0, ..., a_r)$. We divide both sides by p^t and let $g(x) := \sum_{i\in[0,r]} \frac{a_i}{p^t} \cdot x^i$. If $f(\alpha) = 0$, then $g(\alpha) = 0 \mod p$. There are at most r roots $\alpha_0 \in \mathbb{F}_{p^d}$ for polynomial $g(\alpha_0) = 0 \mod p$. Therefore, there are at most $rp^{(k-1)d}$ roots satisfying $f(\alpha) = 0$.*

Lemma 6 (Lemma 2, restated). *Let (ϕ, ψ) be a (m, d)-RMFE over Galois ring $\mathtt{GR}(p^k, r)$, we have that $\mathtt{GR}(p^k, rd) = \phi(\mathbf{1}) \cdot \mathrm{Im}(\phi) \oplus \mathrm{Ker}(\psi)$.*

Proof. *As ϕ is injective and ψ is surjective, ψ induces a bijection from the set $\phi(\mathbf{1}) \cdot \mathrm{Im}(\phi)$ to $\mathtt{GR}(p^k, r)^m$ by definition. Thus, the Galois ring $\mathtt{GR}(p^k, rd)$ is the direct sum of $\phi(\mathbf{1}) \cdot \mathrm{Im}(\phi)$ and $\mathrm{Ker}(\psi)$.*

Lemma 7 (Lemma 3, restated). *There always exists a (m, d)-RMFE over Galois ring $\mathtt{GR}(p^k, r)$ with $\phi(\mathbf{1}) = 1$.*

Proof. *Let (ϕ, ψ) be a (m, d)-RMFE over $\mathtt{GR}(p^k, r)$. Assume $\phi(\mathbf{1})$ is a zero divisor in $\mathtt{GR}(p^k, r)$, $p^{k-1} \cdot \phi(\mathbf{1}) = \phi(p^{k-1}) = 0$, which makes a contradiction since ϕ is injective. Thus, $\phi(\mathbf{1})$ is invertible. Define $\phi' : \mathtt{GR}(p^k, r)^m \mapsto \mathtt{GR}(p^k, rd)$ as $\phi'(\boldsymbol{a}) := \phi(\boldsymbol{a} \cdot \phi(\mathbf{1})^{-1})$ for $\boldsymbol{a} \in \mathtt{GR}(p^k, r)^m$ and $\psi' : \mathtt{GR}(p^k, rd) \mapsto \mathtt{GR}(p^k, r)^m$ as $\psi'(b) := \psi(b \cdot \phi(\mathbf{1}))$ for $b \in \mathtt{GR}(p^k, rd)$. It can be verified that (ϕ', ψ') is a (m, d)-RMFE over $\mathtt{GR}(p^k, r)$ satisfying $\phi(\mathbf{1}) = 1$. This completes the proof.*

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

Input. Upon receiving $(\mathtt{input}, \mathtt{id}, w)$ from a prover $\mathcal{P}$ and $(\mathtt{input}, \mathtt{id})$ from a verifier $\mathcal{V}$, with $\mathtt{id}$ a fresh identifier, store $(\mathtt{id}, w)$.

Prove circuit satisfiability. Upon receiving $(\mathtt{prove}, \mathcal{C}, \mathtt{id}_1, ..., \mathtt{id}_{mn})$ from $\mathcal{P}$ and $(\mathtt{verify}, \mathcal{C}, \mathtt{id}_1, ..., \mathtt{id}_{mn})$ from $\mathcal{V}$ where $\mathtt{id}_1, ..., \mathtt{id}_{mn}$ are present in memory, retrieve $(\mathtt{id}_i, w_i)$ for $i \in [mn]$ and define vectors $\boldsymbol{w_j} := (w_{n(j-1)+1}, ..., w_{nj})$, $j \in [m]$. Send $\mathtt{true}$ to $\mathcal{V}$ if $\mathcal{C}(\boldsymbol{w_j}) = 1$ for all $j \in [m]$ and $\mathtt{false}$ otherwise.

Fig. 12: Zero-knowledge functionality for circuit satisfiability.

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

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

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

Fig. 13: Ideal functionality for equality tests.

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

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. 14: Ideal functionality for oblivious transfer.

Functionality $\mathcal{F}_{\text{vOLE}}^{\mathcal{R}}$

Parameterized by a ring $\mathcal{R}$, length parameters $l_1, ..., l_n \in \mathbb{N}$.

Setup phase: Upon receiving input $(sid; \alpha)$ from $P_{\mathcal{R}}$ where $\alpha \in \mathcal{R}$ and sid is a session identifier, store $(sid; \alpha)$, send $(sid; \texttt{initialized})$ to the adversary and ignore any further inputs from $P_{\mathcal{R}}$ with the same session identifier sid.
Send phases: Upon receiving input $(sid; \boldsymbol{a}; \boldsymbol{b}; i)$ from $P_{\mathcal{S}}$ where $(\boldsymbol{a}, \boldsymbol{b}; i) \in \mathcal{R}^{l_i} \times \mathcal{R}^{l_i} \times \mathbb{N}$ and sid is a session identifier, store $(sid; \boldsymbol{a}; \boldsymbol{b}; i)$, send $(sid; \texttt{sent}; i)$ to the adversary, and ignore any further inputs from $P_{\mathcal{S}}$ with the same session identifier sid and the same value of i.
Deliver phases: Upon receiving a message $(sid; \texttt{Delivery}; i)$ from the adversary where $i \in \mathbb{N}$ and sid is a session identifier, verify that there are stored inputs $(sid; \alpha)$ from $P_{\mathcal{R}}$ and $(sid; \boldsymbol{a}, \boldsymbol{b}; i)$ from $P_{\mathcal{S}}$; else ignore that message. Next, compute $\boldsymbol{v} := \boldsymbol{a} \cdot \alpha + \boldsymbol{b}$, send $(sid; \boldsymbol{v}; i)$ to $P_{\mathcal{R}}$, and ignore further messages $(sid; \texttt{Delivery}; i)$ from the adversary with the same session identifier sid and the same value of i.

Fig. 15: Ideal functionality for fixed input VOLE over $\mathcal{R}$.

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

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

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

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

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

(1) If $P_{\mathcal{R}}$ is honest, sample $\boldsymbol{v} \xleftarrow{\$} \mathtt{GR}(2^k, d)^n$. Otherwise receive $\boldsymbol{v}$ from $\mathcal{A}$.

(2) If $P_{\mathcal{S}}$ is honest, sample uniform $\boldsymbol{u}$ from $\mathtt{GR}(2^k, d)^n$ with exactly one entry invertible and zeros everywhere else, and compute $\boldsymbol{w} := \boldsymbol{v} - \Delta \cdot \boldsymbol{u} \in \mathtt{GR}(2^k, d)$. Otherwise, receive $\boldsymbol{u} \in \mathtt{GR}(2^k, d)$ (with at most one nonzero entry) and $\boldsymbol{w} \in \mathtt{GR}(2^k, d)$ from $\mathcal{A}$, then recompute $\boldsymbol{v} := \boldsymbol{w} + \Delta \cdot \boldsymbol{u}$.

(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 $\boldsymbol{u}$. If $\alpha \in I$, send `success` to $P_{\mathcal{R}}$ and continue. Otherwise, send `abort` to both parties and abort.

(4) Send $(\boldsymbol{u}, \boldsymbol{w})$ to $P_{\mathcal{S}}$ and $\boldsymbol{v}$ to $P_{\mathcal{R}}$.

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

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

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

(1) If $P_{\mathcal{R}}$ is honest, sample $\boldsymbol{K}, \boldsymbol{a} \xleftarrow{\$} \mathtt{GR}(2^k, d)^t$. Otherwise, receive $\boldsymbol{K}, \boldsymbol{a}$ from $\mathcal{A}$.

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

(3) Send $(\boldsymbol{M}, \boldsymbol{c}, \boldsymbol{y})$ to $P_{\mathcal{S}}$ and send $(\boldsymbol{K}, \boldsymbol{a})$ to $P_{\mathcal{R}}$.

Fig. 18: Ideal functionality for generalized VOLE over $\mathtt{GR}(2^k, d)$.

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

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

MP-Extend: Upon receiving $(\mathtt{MP\text{-}Extend}, n = t \cdot 2^h)$ from both parties, proceed as follows:

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

Fig. 19: Ideal functionality for multi-point VOLE over $\mathtt{GR}(2^k, d)$.

Algorithm $\texttt{GGM}$

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

1. $\texttt{GGM.KeyGen}$: on input 1^κ, output $s \xleftarrow{\$} \{0,1\}^\kappa$.
2. $\texttt{GGM.Gen}$: 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 $(\{v_j\}_{j\in[0,n)}, \{(K_{0,i}, K_{1,i})\}_{i\in[h]})$, $\{\Gamma_j\}_{j\in[0,2^{h-1})}$.
3. $\texttt{GGM.Eval}$: 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 $(\{v_j\}_{j\in[0,n)\backslash\{\alpha\}})$.
4. $\texttt{GGM.Eval}'$: 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. 20: Algorithms for GGM tree based PPRF.

B LPN over Galois Rings

B.1 Learning Parity with Noise

The Learning Parity with Noise (LPN) assumption [6] has been studied for decades, and it was adapted to be defined over a finite field $\mathbb{F}_q$ [7] or an integer ring $\mathbb{Z}_n$ [3], 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 $t \leq n$, $\mathtt{Im}(\mathcal{D}_{t,n}(\mathcal{R})) \subset \mathcal{R}^n$. 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*

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

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

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

Definition 4 (Dual LPN). *Let $\mathcal{D}(\mathcal{R}) = \{\mathcal{D}_{t,n}(\mathcal{R})\}$ be the family of distributions over the ring $\mathcal{R}$ where for any integers $t \leq n$, $\mathtt{Im}(\mathcal{D}_{t,n}(\mathcal{R})) \subset \mathcal{R}^n$. Let $\mathcal{G}^\perp$ be a probabilistic dual code generation algorithm such that $\mathcal{G}^\perp(m, n, \mathcal{R})$ outputs a parity check matrix $H \in \mathcal{R}^{n\times(n-m)}$. Let parameters m, n, t be implicit functions of security parameter 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*

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

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

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. $\boldsymbol{e} \leftarrow \mathtt{Ber}_{\lambda,n}(\mathcal{R})$ indicates that each component of $\boldsymbol{e}$ is a uniform random element in $\mathcal{R}$ with probability λ and zero with probability $1 - \lambda$. Thus the expected Hamming weight of $\boldsymbol{e}$ is $\lambda N(|\mathcal{R}| - 1)/|\mathcal{R}|$. We remark that this definition is equivalent to sampling a uniform non-zero element in $\mathcal{R}$ with probability $\lambda N(|\mathcal{R}| - 1)/|\mathcal{R}|$ for each component.

Fixed Hamming weight. Let $\mathtt{HW}(\mathcal{R}) = \{\mathtt{HW}_{t,n}(\mathcal{R})\}_{t,n}$ be the family of distributions of uniform fixed-weight vectors. Let H denote the set $\{\boldsymbol{e} \in \mathcal{R}^n \mid \mathtt{wt}(\boldsymbol{e}) = t\}$. Thus we have $\boldsymbol{e} \leftarrow \mathtt{HW}_{t,n}(\mathcal{R}) \iff \boldsymbol{e} \leftarrow \mathrm{U}_H$.

Regular Hamming weight. Let $\mathtt{RG}(\mathcal{R}) = \{\mathtt{RG}_{t,n}(\mathcal{R})\}_{t,n}$ be the family of distributions of uniform regular weight vectors. W.o.l.g. we assume $t|n$ and let $\boldsymbol{e} = (\boldsymbol{e}_1, ..., \boldsymbol{e}_t)$, where $\boldsymbol{e}_i \in \mathcal{R}^{n/t}, i \in [t]$. Let G denote the set $\{\boldsymbol{e} \in \mathcal{R}^n \mid \mathtt{wt}(\boldsymbol{e}_i) = 1, i \in [t]\}$. Thus we have $\boldsymbol{e} \leftarrow \mathtt{RG}_{t,n}(\mathcal{R}) \iff \boldsymbol{e} \leftarrow \mathrm{U}_G$, and $\mathtt{RG}(\mathcal{R})$ can be viewed as a special case of $\mathtt{HW}(\mathcal{R})$.

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

We adapt reductions for LPN over $\mathbb{Z}_{2^k}$ [35] to Galois rings, obtaining the following theorems.

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

Proof. *Let* $(A, \boldsymbol{b} := \boldsymbol{x} \cdot A + \boldsymbol{e})$ *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* $(A^0 := A \mod 2, \boldsymbol{b}^0 := \boldsymbol{b} \mod 2)$ *constitute exactly the LPN samples over* $\mathbb{F}_{2^d}$ *for noise* $\boldsymbol{e}^0 := \boldsymbol{e} \mod 2$. *Since* $\boldsymbol{e} \leftarrow \mathtt{HW}_{t,n}(\mathtt{GR}(2^k, d))$, *the noise vector* $\boldsymbol{e}^0$ *follows a Bernoulli-like distribution over* $\mathbb{F}_{2^d}$. *Further, it can be observed that* $\boldsymbol{e}^0$ *has expected weight* $t' = \frac{2^{d(k-1)}(2^d-1)}{2^{dk}-1} \cdot t$. *One can naturally generalize Lemma 2 in [35] and obtain that the noise vector* $\boldsymbol{e}^0$ *follows the uniform fixed-weight distribution* $\mathtt{HW}_{t',n}(\mathbb{F}_{2^d})$ *with probability* $\Omega\big(1/\sqrt{t}\big)$. *On the other hand,* $(A^0, \boldsymbol{u}^0)$ *are uniformly random as well. Therefore, one can use a* $(\mathtt{HW}, \mathcal{G}, \mathbb{F}_{2^d}) - \mathrm{LPN}(m, n, t')$ *distinguisher to distinguish* $(A^0, \boldsymbol{b}^0)$ *from uniform samples. The proof for the second statement is similar, except that for* $\boldsymbol{e} \leftarrow \mathtt{Ber}_{\lambda,n}(\mathtt{GR}(2^k, d))$, *one can immediately gets that* $\boldsymbol{e}^0 \sim \mathtt{Ber}_{\lambda,n}(\mathbb{F}_{2^d})$.

Theorem 10. *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* $(A, \boldsymbol{b} := \boldsymbol{x} \cdot A + \boldsymbol{e})$ *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 $a_i \in \mathbb{F}_{2^d}, i = 0, 1, ..., k-1$. *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* $(A^0, A^1, ..., A^{k-1}) := \mathtt{Decom}(A)$, $(\boldsymbol{x}^0, \boldsymbol{x}^1, ..., \boldsymbol{x}^{k-1}) := \mathtt{Decom}(\boldsymbol{x})$, $(\boldsymbol{e}^0, \boldsymbol{e}^1, ..., \boldsymbol{e}^{k-1}) := \mathtt{Decom}(\boldsymbol{e})$, $(\boldsymbol{b}^0, \boldsymbol{b}^1, ..., \boldsymbol{b}^{k-1}) := \mathtt{Decom}(\boldsymbol{b})$. *Therefore, we have* $\boldsymbol{b}^0 = \boldsymbol{x}^0 \cdot A^0 + \boldsymbol{e}^0$, *and for* $i = 1, ..., k-1$, *we have that*

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

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

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

where $\boldsymbol{u}_i \xleftarrow{\$} \mathbb{F}_{2^d}^n$ *for* $i = 0, 1, ..., k-1$. *Since* $\boldsymbol{x}$ *is sampled uniformly at random, its decomposition* $\boldsymbol{x}^i$ *are independent and uniformly random as well. To distinguish the adjacent hybrids* H_i *and* H_{i+1}, *it suffices to distinguish* $\boldsymbol{u}_i$ *and* $\boldsymbol{b}^i$ *with the*

knowledge of $(\boldsymbol{b}^0, ..., \boldsymbol{b}^{i-1})$*. The* f_i *term can be neglected if the effective noise rate of* $\boldsymbol{e}^i$ *conditioned on* $\boldsymbol{e}^0, ..., \boldsymbol{e}^{i-1}$ *is sufficient to make* $\boldsymbol{b}^i$ *pseudorandom. Consider a single noise sample* $(e_j^0, e_j^1, ..., e_j^{k-1}) \leftarrow \mathtt{Decom}(\mathtt{Ber}_{\lambda,n}(\mathtt{GR}(2^k, d)))$*, where* e_j^i *is the* j*-th entry of* $\boldsymbol{e}^i$*. If there exists a non-zero component in* $(e_j^0, e_j^1, ..., e_j^{i-1})$*,* e_j^i *must be uniformly random, which makes* b_j^i *uniformly random. Thus, we have that*

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

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

B.3 Attacks on LPN over Galois Rings

To avoid the above reduction attack, one can let the errors be invertible in $\mathtt{GR}(2^k, d)$ when instantiating LPN instances over $\mathtt{GR}(2^k, d)$, although it only reduces approximately $1/2^d$ fraction of noise rate. We are not aware of any advantages for attacks over Galois rings compared to Galois fields. We review the main attacks that may work on LPN over Galois rings, following the analysis of [24,7,8,3,35].

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 [7]. 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[7]	Estimation(ours)
108112	358620	215	158.15	140.36	158.11
148912	649590	295	158.96	145.71	158.93

Statistical Decoding (SD) [20] In [7], they simplified the complexity of this attack by $\log(m+1) + t \cdot \log \frac{n}{n-m-1}$. A recent work [35] pointed out that to succeed with constant advantage, SD attack requires

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

bits arithmetic operations over $\mathbb{F}_q$. Their result can be naturally adapted to Galois rings, as a Galois ring $\mathtt{GR}(p^k, d)$ contains a field $\mathbb{F}_{p^d}$. We obtain that the bit security of the LPN instance with respect to SD attack is computed as

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

Information Set Decoding (ISD) In [35], they analysed known ISD variants for different field size and show that the generalized SD-ISD attack [38] is equivalent to 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 for our constructions, we use the pooled Gaussian attack to estimate the LPN security.

B.4 Definitions for LPN variants

Recall that our basic construction of VOLE allows the adversary to guess a subset that contains the noisy position for each block. Generally, this would leak 1-bit information: the adversary learns whether his guesses are all right. We describe the corresponding LPN security game G_{LPN} as follows:

1. Let $A \leftarrow \mathcal{G}(m, n) \in \mathtt{GR}(2^k, d)^{m \times n}$ be a primal LPN matrix, $\boldsymbol{x} \leftarrow \mathtt{GR}(2^k, d)^m$ be the secret, and $\boldsymbol{e} = (\boldsymbol{e_1}, ..., \boldsymbol{e_t}) \in \mathtt{GR}(2^k, d)^n$ be the error vector, where each $\boldsymbol{e_i}$ 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 $\boldsymbol{e_i}$, send $\mathtt{success}$ to $\mathcal{A}$. Otherwise, abort.

3. Pick $b \leftarrow \{0, 1\}$. If $b = 0$, send $\boldsymbol{y} := \boldsymbol{x} \cdot A + \boldsymbol{e}$ to $\mathcal{A}$, otherwise, send $\boldsymbol{y} \xleftarrow{\$} \mathtt{GR}(2^k, d)^n$ to $\mathcal{A}$.

4. $\mathcal{A}$ outputs a bit b'. The game outputs 1 if $b' = b$, and outputs 0 otherwise.

Thus, we give the formal definition for the variant of the LPN assumption used in our basic VOLE construction.

Definition 5 (Primal LPN with static leakage). *We say that the decisional* $(\mathtt{RG}, \mathcal{G}, \mathcal{R})-\mathrm{LPN}(m, n, t)$ *with static leakage is* (T, ϵ)*-hard, if for every probabilistic distinguisher* $\mathcal{B}$ *running in time* T*,* $\mathcal{B}$ *wins the game* G_{LPN} *with advantage at most* ϵ*, i.e.*

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

Notice that our second two-round variant allows the adversary to do a second guess. Similarly, this would introduce average one more bit leakage on the noisy positions. We 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 $\boldsymbol{e} = (\boldsymbol{e_1}, ..., \boldsymbol{e_t}) \in \mathtt{GR}(2^k, d)^n$ be the error vector, where each $\boldsymbol{e_i}$ 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 $\boldsymbol{e_i}$, 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 $\boldsymbol{e_i}$, send `success` to $\mathcal{A}$. Otherwise, abort.

3. Pick $b \leftarrow \{0, 1\}$. If $b = 0$, send $\boldsymbol{y} := \boldsymbol{e} \cdot H$ to $\mathcal{A}$, otherwise, send $\boldsymbol{y} \xleftarrow{\$} \mathtt{GR}(2^k, d)^{(n-m)}$ to $\mathcal{A}$.

4. $\mathcal{A}$ outputs a bit b'. The game outputs 1 if $b' = b$, and outputs 0 otherwise.

Thus, we give the formal definition for the variant of the dual LPN assumption used in our non-interactive VOLE construction.

Definition 6 (Dual LPN with static leakage). *We say that the decisional dual* $(\mathtt{RG}, \mathcal{G}, \mathcal{R}) - \mathrm{LPN}(m, n, t)$ *with static leakage is* (T, ϵ)*-hard, if for every probabilistic distinguisher* $\mathcal{B}$ *running in time* T*,* $\mathcal{B}$ *wins the game* G_{Dual} *with advantage at most* ϵ*, i.e.*

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

We assume that the above two kinds of leakage would not decrease the security of primal (dual) LPN, respectively. Thus, we select LPN parameters according to the pooled Gaussian Elimination attack.

C Missing Proofs of Zero-Knowledge Protocols

C.1 Security of re-embedding VOLE

Theorem 11 (Theorem 1, restated). $\Pi_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$ *UC-realizes* $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$ *in the* $(\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}, \mathcal{F}_{\mathrm{EQ}})$*-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* $2^{-s} + 2^{-d}$*.*

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 functionalities* $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}, \mathcal{F}_{\mathrm{EQ}}$*.*

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

1. $S_{\mathcal{S}}$ *reads* $\boldsymbol{M}, \boldsymbol{x} \in \mathtt{GR}(2^k, d)^{n+s}$ *that* $\mathcal{A}$ *sends to* $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$*.*

2. Upon receiving $\boldsymbol{\eta} \in \mathtt{GR}(2^k, d)^{n+s}$ *from* $\mathcal{A}$*, if* $\boldsymbol{\eta} \notin \mathtt{Ker}(\psi)^{n+s}$*,* $S_{\mathcal{S}}$ *aborts.* $S_{\mathcal{S}}$ *sets* $\boldsymbol{M}' := \boldsymbol{M}$*.*

3. $S_{\mathcal{S}}$ *samples* $\boldsymbol{\chi}^1, ..., \boldsymbol{\chi}^s \xleftarrow{\$} \mathbb{Z}_{2^k}^n$*, and sends them to* $\mathcal{A}$*.*

4. Upon receiving $\boldsymbol{a}, \boldsymbol{b} \in \mathtt{GR}(2^k, d)^s$ *from* $\mathcal{A}$*. If* $b_i - a_i \neq \eta_{n+i} + \sum_{j \in [n]} \chi_j^i \cdot \eta_j$*, for some* $i \in [s]$*, or* $\boldsymbol{b} \notin \mathtt{Im}(\phi)^s$*,* $S_{\mathcal{S}}$ *aborts.*

5. $S_{\mathcal{S}}$ *reads* $\hat{M}_i, i \in [s]$ *that* $\mathcal{A}$ *sends to* $\mathcal{F}_{\mathrm{EQ}}$*.* $S_{\mathcal{S}}$ *computes* $\hat{M}_i' := M_{n+i} + \sum_{j \in [n]} \chi_j^i \cdot M_j$*, for* $i \in [s]$*. If both* $\boldsymbol{\eta} = (\tau(\boldsymbol{x}) - \boldsymbol{x})$ *and* $\hat{\boldsymbol{M}} = \hat{\boldsymbol{M}'}$ *hold,* $S_{\mathcal{S}}$

sends `true` *to* $\mathcal{A}$ *(emulating* $\mathcal{F}_{\mathrm{EQ}}$*), and sends* $(\boldsymbol{x}[1:n], \boldsymbol{M}[1:n])$ *to* $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$*. Otherwise,* $S_{\mathcal{S}}$ *sends* `false` *to* $\mathcal{A}$ *and 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 simulation for* $\mathcal{F}_{\mathrm{EQ}}$*. Basically,* $\mathcal{A}$ *can cheat by sending* $\boldsymbol{\eta} \in \mathtt{Ker}(\psi)^{n+s}$*, but* $\boldsymbol{\eta} \neq \tau(\boldsymbol{x}) - \boldsymbol{x}$*. Let* $\boldsymbol{\eta} = \tau(\boldsymbol{x}) - \boldsymbol{x} + \boldsymbol{\varepsilon}$*, where* $\boldsymbol{\varepsilon} \in \mathtt{Ker}(\psi)^{n+s}$*. We have that*

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

Therefore, $\mathcal{A}$ *can set*

$$a_i^* := (x_{n+i} - \varepsilon_{n+i}) + \sum_{j\in[n]} \chi_j^i \cdot (x_j - \varepsilon_j),$$

and

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

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

$$\begin{aligned}
\hat{M}_i' &= K_{n+i} + \sum_{j\in[n]} \chi_j^i \cdot K_j - \Delta \cdot a_i^* \\
&= (M_{n+i} + \Delta x_{n+i}) + \sum_{j\in[n]} \chi_j^i (M_j + \Delta x_j) - \Delta(x_{n+i} - \varepsilon_{n+i} + \sum_{j\in[n]} \chi_j^i (x_j - \varepsilon_j)) \\
&= M_{n+i} + \sum_{j\in[n]} \chi_j^i \cdot M_j + \Delta \cdot (\varepsilon_{n+i} + \sum_{j\in[n]} \chi_j^i \cdot \varepsilon_j) \\
&= \hat{M}_i + \Delta \cdot (\varepsilon_{n+i} + \sum_{j\in[n]} \chi_j^i \cdot \varepsilon_j).
\end{aligned}$$

Thus, $\mathcal{F}_{\mathrm{EQ}}$ *returns* `true` *if and only if*

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

for all $i \in [s]$*. If* $\varepsilon_{n+i} + \sum_{i\in[n]} \chi_j^i \cdot \varepsilon_j \neq 0$*, from lemma 1, the above equality holds with probability at most* 2^{-d}*. Since* $\chi_j^i \in \mathbb{Z}_{2^k}$*, for* $j \in [n]$*,* $\varepsilon_{n+i} + \sum_{j\in[n]} \chi_j^i \cdot \varepsilon_j = 0$ *holds with probability at most* $1/2$*. Combining together,* $\mathcal{F}_{\mathrm{EQ}}$ *returns* `true` *with probability at most* $2^{-s} + 2^{-d}$ *in real execution if* $\boldsymbol{\eta} \neq \tau(\boldsymbol{x}) - \boldsymbol{x}$*, 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* $\Delta \in \mathtt{GR}(2^k, d)$ *that* $\mathcal{A}$ *sends to* $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$ *in the* `Init` *procedure.* $S_{\mathcal{R}}$ *then sends* Δ *to* $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$*. Every time the* ***`Extend-pair`*** *procedure is executed,* $S_{\mathcal{R}}$ *does as follows:*

1. $S_{\mathcal{R}}$ records $\boldsymbol{K} \in \mathtt{GR}(2^k,d)^{n+s}$ sent by $\mathcal{A}$. Upon receiving $\boldsymbol{\eta} \in \mathtt{Ker}(\psi)^n$ from $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$, $S_{\mathcal{R}}$ samples $\boldsymbol{\eta}' \xleftarrow{\$} \mathtt{Ker}(\psi)^s$ and sends $\hat{\boldsymbol{\eta}} := (\boldsymbol{\eta}, \boldsymbol{\eta}') \in \mathtt{Ker}(\psi)^{n+s}$ to $\mathcal{A}$.

2. $S_{\mathcal{R}}$ sets $\boldsymbol{K}' := \boldsymbol{K} + \Delta \cdot \hat{\boldsymbol{\eta}}$. Upon receiving $\boldsymbol{\chi}^i \in \mathbb{Z}_{2^k}^n, i \in [s]$ from $\mathcal{A}$, $S_{\mathcal{R}}$ samples $\boldsymbol{b} \xleftarrow{\$} \mathtt{Im}(\phi)^s$, and computes $a_i := b_i - \eta_{n+i} - \sum_{j\in[n]} \chi_j^i \cdot \eta_j$, for $i \in [s]$. Then, $S_{\mathcal{R}}$ sends $\boldsymbol{a}, \boldsymbol{b}$ to $\mathcal{A}$.

3. $S_{\mathcal{R}}$ reads $\hat{\boldsymbol{M}}'$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{EQ}}$. $S_{\mathcal{R}}$ computes $\hat{M}_i := K_{n+i} + \sum_{j\in[n]} \chi_j^i \cdot K_j - \Delta \cdot a_i$, for $i \in [s]$. $S_{\mathcal{R}}$ sends $\hat{\boldsymbol{M}}$ to $\mathcal{A}$ (emulating $\mathcal{F}_{\mathrm{EQ}}$). If $\hat{\boldsymbol{M}}' = \hat{\boldsymbol{M}}$, $S_{\mathcal{R}}$ sends `true` *to $\mathcal{A}$. Otherwise, $S_{\mathcal{R}}$ sends* `abort` *to $\mathcal{A}$ and aborts.*

4. $S_{\mathcal{R}}$ sends $\boldsymbol{K}[1:n]$ 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 $\boldsymbol{\eta} \in \mathtt{Ker}(\psi)^{n+s}$. Since $\boldsymbol{x}$ is distributed uniformly at random in $\mathtt{GR}(2^k,d)^{n+s}$, $\boldsymbol{\eta}$ is distributed uniformly at random in $\mathtt{Ker}(\psi)^{n+s}$. While in simulation, $\hat{\boldsymbol{\eta}}$ are uniformly sampled from $\mathtt{Ker}(\psi)^{n+s}$ as well. As for $(\boldsymbol{a}, \boldsymbol{b} = \tau(\boldsymbol{a}))$ in real protocol, $\boldsymbol{a}$ is masked with $\boldsymbol{x}[n+1:n+s]$, which makes $\boldsymbol{b}$ have the uniform distribution on $\mathtt{Im}(\phi)^s$. Thus, $(\boldsymbol{a}, \boldsymbol{b})$ generated by $S_{\mathcal{R}}$ have the same distribution as that in the real protocol. The final message that $\mathcal{A}$ receives from $\mathcal{F}_{\mathrm{EQ}}$ is $\hat{\boldsymbol{M}}$. It can be easily verified that

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

Thus $\hat{\boldsymbol{M}}$ has the same distribution in both worlds. Further, if $\mathcal{F}_{\mathrm{EQ}}$ aborts in real execution, $S_{\mathcal{R}}$ will send `false` *to $\mathcal{A}$ and aborts as well. Finally, we turn to the output of the honest $P_{\mathcal{S}}$. In real protocol, x_i is conditioned on that $\tau(x_i) - x_i = \eta_i$, 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 Basic Construction

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 $(\boldsymbol{\mu}, \boldsymbol{M_\mu}, \boldsymbol{M'_\mu})$, $(\boldsymbol{\nu}, \boldsymbol{M_\nu}, \boldsymbol{M'_\nu})$ and (π, M_π) that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$. Thus, $\mathcal{S}$ can immediately obtain the MACs $([\mu_i], [\tau(\mu_i)])_{i\in[n]}$, $([\nu_i], [\tau(\nu_i)])_{i\in[t]}$, $[\pi]$.

2. Upon receiving $\boldsymbol{\delta}$ from $\mathcal{A}$, $\mathcal{S}$ 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 $(\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_m) := \psi(\tau(\boldsymbol{\omega}))$ and sets $\boldsymbol{w} := (\boldsymbol{w}_1^\top, ..., \boldsymbol{w}_m^\top) \in \mathbb{Z}_{2^k}^{mn}$, then $\mathcal{S}$ sends $\boldsymbol{w}$ and the circuit $\mathcal{C}$ to $\mathcal{F}_{\text{ZK}}^m$. Otherwise, $\mathcal{S}$ sends $\boldsymbol{w} := \perp$ and $\mathcal{C}$ to $\mathcal{F}_{\text{ZK}}^m$ 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 $\mathcal{A}$ have to prepare a $\boldsymbol{\omega} \in \texttt{Im}(\phi)^n$, 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 $\Pi_{\text{ZK}}^{m,n,k}$. 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 $e_i \in \texttt{GR}(2^k, d)$. Thus we have that*

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

and

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

which leads to

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

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

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

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

Finally, we show that if $\mathcal{C}(\boldsymbol{w}_i) = 0$, for some $i \in [m]$, and all the values on the wires in the circuit are correct, the probability that $\mathcal{A}$ successfully provides a $M'_{\omega_h} := M_{\omega_h} + e_{\omega_h}$ such that $K_{\omega_h} = M'_{\omega_h} + \Delta \cdot \phi(\mathbf{1})$ is upper bounded by 2^{-d}. Let $\boldsymbol{r} := (\mathcal{C}(\boldsymbol{w}_1), ..., \mathcal{C}(\boldsymbol{w}_m)) \in \{0,1\}^m$. After executing the protocol, We have

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

Thus, honest verifier accepts if and only if

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

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

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

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

1. In the offline phase: $\mathcal{S}$ records $\Delta \in \mathtt{GR}(2^k, d)$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$ in the `Init` procedure, also, $\mathcal{S}$ records $(\boldsymbol{K_\mu}, \boldsymbol{K'_\mu}), (\boldsymbol{K_\nu}, \boldsymbol{K'_\nu})$ and K_π that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$. Besides, $\mathcal{S}$ samples $\tau(\boldsymbol{\mu}) - \boldsymbol{\mu} \xleftarrow{\$} \mathtt{Ker}(\psi)^n$ and $\tau(\boldsymbol{\nu}) - \boldsymbol{\nu} \xleftarrow{\$} \mathtt{Ker}(\psi)^t$, and sends them to $\mathcal{A}$.

2. $\mathcal{S}$ samples $\tau(\boldsymbol{\delta}) \xleftarrow{\$} \mathtt{Im}(\phi)^n$ and sets $\boldsymbol{\delta} := \tau(\boldsymbol{\delta}) + \tau(\boldsymbol{\mu}) - \boldsymbol{\mu}$. $\mathcal{S}$ sends $\boldsymbol{\delta}$ to $\mathcal{A}$. $\mathcal{S}$ computes $K_{\tau(\omega_i)} := K'_{\mu_i} + \Delta \cdot \tau(\delta_i)$, for $i \in [n]$. Besides, $\mathcal{S}$ samples $\boldsymbol{\omega} \xleftarrow{\$} \mathtt{Im}(\phi)^n$.

3. For each gate $(\alpha, \beta, \gamma, T) \in \mathcal{C}$, in a topological order:

- If T=`Add`, $\mathcal{S}$ computes $K_{\omega_\gamma} := K_{\omega_\alpha} + K_{\omega_\beta}$ as the honest $\mathcal{V}$ would do, and sets $\omega_\gamma := \omega_\alpha + \omega_\beta$.

- If T=`Mul`, and this is the i-th multiplication gate, then $\mathcal{S}$ sends $d_i := \omega_\alpha \cdot \omega_\beta - \nu_i$ to $\mathcal{A}$. $\mathcal{S}$ computes $K_{\omega_\gamma} := K'_{\nu_i} + \Delta \cdot \tau(d_i)$, $K_{\hat{\omega}_\gamma} := K_{\nu_i} + \Delta \cdot d_i$ and $B_i := K_{\omega_\alpha} \cdot K_{\omega_\beta} - \Delta \cdot K_{\hat{\omega}_\gamma}$ as the honest verifier would do, and sets $\omega_\gamma := \tau(\omega_\alpha \cdot \omega_\beta)$.

4. $\mathcal{S}$ receives $\boldsymbol{\chi} \in \mathbb{Z}_{2^k}^t$ from $\mathcal{A}$.

5. $\mathcal{S}$ computes $Z := \sum_{i\in[t]} \chi_i \cdot B_i + B^* \in \mathtt{GR}(2^k, d)$, where $B^* := K_\pi$. Then $\mathcal{S}$ samples $Y \xleftarrow{\$} \mathtt{GR}(2^k, d)$ and sets $X := Z - \Delta \cdot Y$. $\mathcal{S}$ sends (X, Y) to $\mathcal{A}$.

6. For the single output wire ω_h, $\mathcal{S}$ already holds K_{ω_h}. $\mathcal{S}$ computes $M_{\omega_h} := K_{\omega_h} - \Delta \cdot \phi(\mathbf{1})$, and sends M_{ω_h} to $\mathcal{A}$.

Since $\mathtt{GR}(2^k, d) = \mathtt{Im}(\phi) \oplus \mathtt{Ker}(\psi)$, 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 $\mathtt{Im}(\phi)^n$. Further, $\boldsymbol{\delta}$ is sufficient to perfectly hide the circuit inputs $\boldsymbol{\omega} := \phi(W)$ in $\mathtt{Im}(\phi)^n$. Similarly, d_i perfectly hides the output value of i-th $\mathtt{Mul}$ gate in $\mathtt{Im}(\phi)$. 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 Sublinear Communication Variant

Protocol Π_{PAC}

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

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

Poly-Key: On input m:

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

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

1. For the input MACs $[\boldsymbol{u}]$, suppose $\mathcal{P}$ holds $\boldsymbol{u}, \boldsymbol{w} \in \mathtt{GR}(2^k, d)^n$ and $\mathcal{V}$ holds $\boldsymbol{w} \in \mathtt{GR}(2^k, d)^n$ such that $\boldsymbol{w} := \boldsymbol{v} - \Delta \cdot \boldsymbol{u}$.
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 $\langle b_j \rangle := \sum_{i \in [m]} f_{j,i} \cdot \langle \Lambda^i \rangle + f_{j,0} - u_j$.
3. $\mathcal{P}$ computes $(\mathtt{com}_2, \mathtt{unv}_2) \leftarrow \mathtt{Commit}(\langle b_1 \rangle, ..., \langle b_n \rangle)$, and sends $\mathtt{com}_2$ to $\mathcal{V}$.

Gen:

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

Fig. 21: Protocol for generating PACs over $\mathtt{GR}(2^k, d)$.

The following theorem shows a reduction from AHE schemes over Galois rings to AHE schemes over Galois fields, which is very similar to that we build reusable VOLE over $\mathtt{GR}(2^k, d)$ from reusable VOLE over $\mathbb{F}_{2^d}$ in Section 4.1.

Theorem 13. *There exists AHE schemes over Galois ring $\mathtt{GR}(p^k, d)$, if there exists AHE schemes over Galois field $\mathbb{F}_{p^d}$. In particular, CPA security, degree restriction and circuit privacy will preserve.*

Proof. *Let $(\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$ be an AHE scheme over $\mathbb{F}_{p^d}$. We denote the encryption of $m \in \mathbb{F}_{p^d}$ by $\langle m \rangle$. $a \in \mathtt{GR}(p^k, d)$ can be uniquely written as $a_0 + a_1 \cdot p + ... + a_{k-1} \cdot p^{k-1}$, where $a_i \in \mathbb{F}_{p^d}$, $i = 0, 1, ..., k-1$. We define the encryption of a as the tuple $(\langle a_0 \rangle, \langle a_1 \rangle, ..., \langle a_{k-1} \rangle)$, denoted by $\lfloor a \rceil$. The addition of two ciphertexts is defined as*

$$\lfloor a \rceil + \lfloor b \rceil := (\langle a_0 \rangle + \langle b_0 \rangle, \langle a_1 \rangle + \langle b_1 \rangle, ..., \langle a_{k-1} \rangle + \langle b_{k-1} \rangle).$$

The scalar multiplication is defined as

$$\begin{aligned}\alpha \lfloor a \rceil :&= (\alpha_0 \langle a_0 \rangle, \alpha_0 \langle a_1 \rangle, ..., \alpha_0 \langle a_{k-1} \rangle) + (0, \alpha_1 \langle a_0 \rangle, ..., \alpha_1 \langle a_{k-2} \rangle) + ... + (0, 0, ..., \alpha_{k-1} \langle a_0 \rangle) \\ &= (\alpha_0 \langle a_0 \rangle, \sum_{i=0}^{1} \alpha_i \langle a_{1-i} \rangle, ..., \sum_{i=0}^{k-1} \alpha_i \langle a_{k-1-i} \rangle).\end{aligned}$$

It can be verified that the above scheme is an AHE over $\mathtt{GR}(p^k, d)$, whose CPA security is straightforward from $(\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$, and one can prove degree restriction and circuit privacy by contradiction.

Theorem 14 (Theorem 3, restated). *Protocol $\Pi_{\mathrm{slZK}}^{m,n,t}$ UC-realizes functionality $\mathcal{F}_{\mathtt{ZK}}^{m}$ 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 $\frac{2m_2+3}{2^d} + \mathsf{negl}(\kappa)$.*

Proof. *Since the proof has the similar structure to that in [42], hence we only explain the difference here. The readers may refer to [42] for more details. Note that for the* `BatchCheck` *procedure, we let $\mathcal{V}$ check that $f(\alpha_i) = \tau(g(\beta_i))$, $i \in [t]$, 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 $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$ which delivers MACs of random re-embedding pairs $(\boldsymbol{\mu}, \tau(\boldsymbol{\mu}))$ over Galois rings. Recall that we let $\mathcal{F}_{\mathrm{embVOLE}}^{\mathtt{GR}(2^k,d)}$ additionally send $\boldsymbol{\mu} - \tau(\boldsymbol{\mu})$ to $\mathcal{V}$, as explained before in the proof of Theorem 2, this leakage would not influence the privacy of witnesses over $\mathbb{Z}_{2^k}$ and it guarantees $\mathcal{S}$ can extract SIMD witnesses correctly if sender is corrupted. Specifically, we instantiate the* DVZK *procedure in [42] by the* LPZK *based approach, namely the multiplication check procedure in protocol $\Pi_{\mathrm{ZK}}^{m,n,t}$. By Lemma 1 and the upper bound in the Galois field case, the soundness error in Galois ring case is upper bounded by $\frac{2m_2+3}{2^d} + \mathsf{negl}(\kappa)$.*

D Missing Proofs of VOLE Protocols

D.1 Security of Basic Construction

Theorem 15 (Theorem 6, 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 $\alpha := \sum_{i\in[h]} 2^{h-i} \cdot a_i$ and defines a vector $\boldsymbol{u} \in \mathtt{GR}(2^k, d)^n$ such that $u_\alpha = \beta$ and $u_i = 0$ for $i \neq \alpha$. Next, $S_{\mathcal{S}}$ runs $\mathtt{GGM.Eval}(\alpha, \{K^i_{\bar{\alpha}_i}\}_{i\in[h]})$ and obtains $\{v_j\}_{j\neq\alpha}$.

3. $S_{\mathcal{S}}$ samples $g \xleftarrow{\$} \mathtt{GR}(2^k, d)$ and sends it to $\mathcal{A}$. Then $S_{\mathcal{S}}$ defines a vector $\boldsymbol{w} \in \mathtt{GR}(2^k, d)^n$ such that $w_\alpha = \delta - (g + \sum_{i\neq\alpha} v_i)$ and $w_i = v_i$ for $i \neq \alpha$.

4. $S_{\mathcal{S}}$ reads the values (x, z) that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$.

5. Upon receiving $\{\chi_i\}_{i\in[0,n)}$ and $x^* \in \mathtt{GR}(2^k, d)$ from $\mathcal{A}$, $S_{\mathcal{S}}$ sets $x' := x^* + x \in \mathtt{GR}(2^k, d)$.

6. $S_{\mathcal{S}}$ reads the values $V_{P_{\mathcal{S}}}$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{EQ}}$. Then $S_{\mathcal{S}}$ computes $V'_{P_{\mathcal{S}}} := \sum_{i\in[0,n)} \chi_i \cdot w_i - z$ and does as follows:

(a) If $x' = \chi_\alpha \cdot \beta$, then $S_{\mathcal{S}}$ checks whether $V_{P_{\mathcal{S}}} = V'_{P_{\mathcal{S}}}$. If so, $S_{\mathcal{S}}$ sends `true` *to $\mathcal{A}$, and sends $\boldsymbol{u}, \boldsymbol{w}$ to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$. Otherwise, $S_{\mathcal{S}}$ sends* `abort` *to $\mathcal{A}$ and aborts.*

(b) If $x' \neq \chi_\alpha \cdot \beta$, since every $a \in \mathtt{GR}(2^k, d)$ can be represented as $a = \sum_{i\in[0,k)} a_i \cdot 2^i, a_i \in \mathbb{F}_{2^d}$. Let id_a denote the least index such that $a_{id_a} \neq 0$ (define $id_0 := k$). It can be observed that a is invertible in $\mathtt{GR}(2^k, d)$ if and only if $id_a = 0$. 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 $\Delta' := \frac{\varepsilon}{2^{id_\eta}} \cdot (\frac{\eta}{2^{id_\eta}})^{-1} \mod 2^{k-id_\eta}$, 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* $\mathcal{F}_{\text{spVOLE}}^{\mathtt{GR}(2^k,d)}$*. Thus, we focus on analysing indistinguishability between the* ***SP-Extend*** *procedures.*

$S_{\mathcal{S}}$ can record a, c that $\mathcal{A}$ sends to $\mathcal{F}_{\text{VOLE}}^{\mathtt{GR}(2^k,d)}$. 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 Δ is uniformly random and β is a unit. Therefore, g sampled uniformly at random by $S_{\mathcal{S}}$ is of the same distribution to that masked with γ in $\mathcal{A}$'s view. Now it remains to consider the check step.

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

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

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

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

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

where $\Delta, \eta, \epsilon \in \mathtt{GR}(2^k, d)$. Although $S_{\mathcal{S}}$ does not hold Δ, he can query the global key to $\mathcal{F}_{\text{spVOLE}}^{\mathtt{GR}(2^k,d)}$. Since the above equation is over $\mathtt{GR}(2^k, d)$, there may be no solutions for Δ, or more than one solutions for Δ. Thus the ***Global-key Query*** *is extended to allow queries of "lower bits" of Δ. It is not hard to see that the simulation matches the real execution in this case.*

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

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

1. $S_{\mathcal{R}}$ reads $b \in \mathtt{GR}(2^k, d)$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$. $S_{\mathcal{R}}$ samples $a' \xleftarrow{\$} \mathtt{GR}(2^k, d)$ and sends a' to $\mathcal{A}$. Then, $S_{\mathcal{R}}$ draws a uniformly random β in the set of units of $\mathtt{GR}(2^k, d)$. Next, $S_{\mathcal{R}}$ computes $\gamma := b + \Delta \cdot a'$ and $\delta := \gamma - \Delta \cdot \beta$.

2. $S_{\mathcal{R}}$ reads the values $(K_0^i, K_1^i)_{i\in[h]}$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{OT}}$. Upon receiving $g \in \mathtt{GR}(2^k, d)$ from $\mathcal{A}$, for each $\alpha \in [0, n)$, $S_{\mathcal{R}}$ defines a vector $\boldsymbol{w}^\alpha$ such that $\{w_j^\alpha\}_{j\neq\alpha} := \mathtt{GGM.Eval}(\alpha, \{K_{\bar{\alpha}_i}^i\}_{i\in[h]})$ and $\boldsymbol{w}_\alpha^\alpha := \delta - (g + \sum_{i\neq\alpha} w_i^\alpha)$.

3. $S_{\mathcal{R}}$ reads the vector y^* that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$. $S_{\mathcal{R}}$ samples $\chi_i \xleftarrow{\$} \mathtt{GR}(2^k, d)$ for $i \in [0, n)$ and $x^* \xleftarrow{\$} \mathtt{GR}(2^k, d)$. $S_{\mathcal{R}}$ sends $(\{\chi_i\}_{i\in[0,n)}, x^*)$ to $\mathcal{A}$. Then, $S_{\mathcal{R}}$ computes $y := y^* + \Delta \cdot x^*$.

4. $S_{\mathcal{R}}$ reads $\mathcal{V}_{P_{\mathcal{R}}}$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{EQ}}$. Then $S_{\mathcal{R}}$ constructs a set $I \subseteq [0, n)$ as follows:

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

(b) Append α satisfying $V_{P_{\mathcal{S}}}^\alpha = V_{P_{\mathcal{R}}}$ to set I, i.e. $I := \{\alpha \in [0, n) \mid V_{P_{\mathcal{S}}}^\alpha = V_{P_{\mathcal{R}}}\}$.

$S_{\mathcal{R}}$ sends I to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$. If it returns $\mathtt{abort}$, $S_{\mathcal{R}}$ samples $\hat{\alpha} \xleftarrow{\$} [0, n)\backslash I$, sends $(\mathtt{false}, V_{P_{\mathcal{S}}}^{\hat{\alpha}})$ to $\mathcal{A}$ (emulating $\mathcal{F}_{\mathrm{EQ}}$), and then aborts. Otherwise, $S_{\mathcal{R}}$ sends $(\mathtt{true}, V_{P_{\mathcal{R}}})$ to $\mathcal{A}$.

5. $S_{\mathcal{R}}$ picks an arbitrary $\alpha \in I$ and defines $\boldsymbol{v}$ such that $v_i := w_i^\alpha$, for $i \neq \alpha$ and $v_\alpha := \gamma - g - \sum_{i\neq\alpha} v_i$. $S_{\mathcal{R}}$ sends $\boldsymbol{v}$ to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$.

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

In real execution, a is uniformly random in $\mathcal{A}$'s view, which perfectly masks a' in $\mathtt{GR}(2^k, d)$. Thus, a' provided by $S_{\mathcal{R}}$ has the same distribution as that in the real execution. $S_{\mathcal{R}}$ learns $\{(K_0^i, K_1^i)\}_{i\in[h]}$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{OT}}$, therefore, $S_{\mathcal{R}}$ can evaluate n punctured GGM trees with one value of the α-th leaf missed, for each $\alpha \in [0, n)$. $S_{\mathcal{R}}$ learns y^* that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$. The indistinguishability of $\boldsymbol{\chi} \in \mathtt{GR}(2^k, d)^n$ is obvious and the next message that $\mathcal{A}$ receives from $P_{\mathcal{S}}$ or $S_{\mathcal{R}}$ is x^*. Similarly, x^* is perfectly masked with x since x is uniformly random in $\mathcal{A}$'s view. Therefore, we obtain the indistinguishability of x^* between two worlds. What remains to consider is the simulation for equality test. The set I constructed by $S_{\mathcal{R}}$ corresponds to the selective failure attack on the α^* of honest $P_{\mathcal{S}}$. This attack can be formalized as follows: (1) $\mathcal{A}$ generates a GGM tree correctly and obtains $(\{v_j\}_{j\in[0,n)}, \{(K_0^i, K_1^i)\}_{i\in[h]})$, (2) $\mathcal{A}$ guesses a set $I \subseteq [0, n)$, (3) Let U denote the union of the sets $\{K_{\bar{\alpha}_i}^i\}_{i\in[h]}$, for $\alpha \in I$. $\mathcal{A}$ keeps the values of U unchanged and randomizes the remaining values in $\{K_0^i, K_1^i\}_{i\in[h]}$. (4) $\mathcal{A}$ runs the rest program as an honest $P_{\mathcal{R}}$. It can be observed that if $\alpha^* \in I$, then the equality check will pass, i.e. $V_{P_{\mathcal{S}}}^{\alpha^*} = V_{P_{\mathcal{R}}}$. This observation guarantees that the set I reconstructed by $S_{\mathcal{R}}$ is identical to that generated by $\mathcal{A}$. Therefore, $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$ 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 $\boldsymbol{v}_{\alpha^*}$ to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$, otherwise, $\mathcal{Z}$ may distinguish two worlds from joint view of $\mathcal{A}$ and output of $P_{\mathcal{S}}$. We claim

that

$$\Pr\Big[\boldsymbol{v}^{\alpha} \neq \boldsymbol{v}^{\alpha'} \;\Big|\; V_{P_{\mathcal{S}}}^{\alpha} = V_{P_{\mathcal{S}}}^{\alpha'}\Big] \leq \frac{1}{2^d}.$$

We have that

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

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

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

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

and

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

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

Theorem 16 (Theorem 7, restated). *If the decisional $(\mathtt{RG}, \mathcal{G}, \mathtt{GR}(2^k,d))$-LPN$(m,n,t)$ with static leakage assumption holds, then $\Pi_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$ UC-realizes $\mathcal{F}_{q\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$ in the $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$-hybrid model.*

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

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

1. For $i \in [t]$, $S_{\mathcal{S}}$ reads $\boldsymbol{e}_i \in \mathtt{GR}(2^k,d)^m$ (with at most one invertible entry and zeros everywhere else) and $\boldsymbol{c}_i \in \mathtt{GR}(2^k,d)^m$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$. Let $\boldsymbol{e} := (\boldsymbol{e}_1, ..., \boldsymbol{e}_t) \in \mathtt{GR}(2^k,d)^n$ and $\boldsymbol{c} := (\boldsymbol{c}_1, ..., \boldsymbol{c}_t) \in \mathtt{GR}(2^k,d)^n$.

2. $S_{\mathcal{S}}$ computes $\boldsymbol{x} := \boldsymbol{u} \cdot A + \boldsymbol{e} \in \mathtt{GR}(2^k,d)^n$ and $\boldsymbol{M} := \boldsymbol{w} \cdot A + \boldsymbol{c} \in \mathtt{GR}(2^k,d)^n$. Then $S_{\mathcal{S}}$ updates $\boldsymbol{u} := \boldsymbol{x}[1:m]$ and $\boldsymbol{z} := \boldsymbol{M}[1:m]$. Next, $S_{\mathcal{S}}$ sends $\boldsymbol{x}[m+1,n]$ and $\boldsymbol{M}[m+1,n]$ to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$.

3. Whenever $\mathcal{A}$ sends a global-key query $(Guess, \Delta', s')$ to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$, $S_{\mathcal{S}}$ sends $(Guess, \Delta')$ to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$ and forwards the answer from $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$ to $\mathcal{A}$. If the answer is `abort`*, $S_{\mathcal{S}}$ aborts.*

The indistinguishability between simulation and execution is obvious, since the outputs of both parties in two worlds are computed in the same way.

Corrupted $P_{\mathcal{R}}$***:*** *$S_{\mathcal{R}}$ first receives $\Delta \in \mathtt{GR}(2^k, d)$ from $\mathcal{A}$ and reads $\boldsymbol{v} \in \mathtt{GR}(2^k, d)^m$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$ in the* `Init` *procedure. $S_{\mathcal{R}}$ sends Δ to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$. Then whenever* `Extend` *procedure is going to run, $S_{\mathcal{R}}$ acts as follows:*

1. For $i \in [t]$, $S_{\mathcal{R}}$ reads the vector $\boldsymbol{b}_i \in \mathtt{GR}(2^k, d)^m$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$. Let $\boldsymbol{b} := (\boldsymbol{b}_1, ..., \boldsymbol{b}_t) \in \mathtt{GR}(2^k, d)^n$.

2. $S_{\mathcal{R}}$ samples a random $\boldsymbol{e} := (\boldsymbol{e}_1, ..., \boldsymbol{e}_t) \in \mathtt{GR}(2^k, d)^n$, where each $\boldsymbol{e}_i$ has one invertible entry and zeros everywhere else. Let $\{\alpha_1, ..., \alpha_t\}$ be the invertible entry in $\{\boldsymbol{e}_1, ..., \boldsymbol{e}_t\}$, respectively. For $i \in [t]$, $S_{\mathcal{R}}$ reads the set $I_i \subseteq [0, m)$ that $\mathcal{A}$ sends to $\mathcal{F}_{\mathrm{spVOLE}}^{\mathtt{GR}(2^k,d)}$. If $\alpha_i \in I_i$, then $S_{\mathcal{R}}$ continues and sends `success` *to $\mathcal{A}$. Otherwise, $S_{\mathcal{R}}$ sends* `abort` *to $\mathcal{A}$ and aborts.*

3. $S_{\mathcal{R}}$ computes $\boldsymbol{K} := \boldsymbol{v} \cdot A + \boldsymbol{b} \in \mathtt{GR}(2^k, d)^n$, and updates $\boldsymbol{v} := \boldsymbol{K}[1 : m]$. $S_{\mathcal{R}}$ sends $\boldsymbol{K}[m + 1 : n]$ to $\mathcal{F}_{\mathrm{VOLE}}^{\mathtt{GR}(2^k,d)}$.

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

D.2 Security of Two-Round Variant

Theorem 17 (Theorem 8, restated). *If G and G' are PRGs with G' having the right half injective property, and $\mathtt{Hash} : \{0,1\}^{2^h\kappa} \mapsto \{0,1\}^\kappa$ is a collision resistant hash function, $\Pi_{\mathrm{mpVOLE}}^{\mathtt{GR}(2^k,d)}$ realizes the functionality $\mathcal{F}_{\mathrm{mpVOLE}}^{\mathtt{GR}(2^k,d)}$ in the $(\mathcal{F}_{\mathrm{gVOLE}}^{\mathtt{GR}(2^k,d)}, \mathcal{F}_{\mathrm{OT}})$-hybrid model with malicious security.*

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

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

2. For $j \in [t]$ and $i \in [h]$, $S_{\mathcal{S}}$ records the input $\bar{\alpha}_i^j$ sent to $\mathcal{F}_{\mathrm{OT}}$ by $\mathcal{A}$, then $S_{\mathcal{S}}$ can recover the values $\alpha^j, j \in [t]$. For $j \in [t]$, $S_{\mathcal{S}}$ computes $s^j \leftarrow \mathtt{GGM.KeyGen}(1^\kappa)$, and runs $\mathtt{GGM.Gen}(2^{h+1}, s^j)$ to obtain $(\{(v_{2i}^j, v_{2i+1}^j, \Gamma_i^j)\}_{i\in[0,2^h)}, \{(K_{0,i}^j, K_{1,i}^j)\}_{i\in[h+1]})$. $S_{\mathcal{S}}$ recomputes $K_{1,h+1}^j := \bigoplus_{i\in[0,2^h)} \Gamma_i^j$. $S_{\mathcal{S}}$ emulates $\mathcal{F}_{\mathrm{OT}}$ by sending $K_{\bar{\alpha}_i^j,i}^j$ to $\mathcal{A}$. $S_{\mathcal{S}}$ sends $g_0^j, g_1^j \xleftarrow{\$} \mathtt{GR}(2^k, d)$ and $K_{1,h+1}^j$ to $\mathcal{A}$.

3. For $j \in [t]$, $S_{\mathcal{S}}$ computes $\Gamma^j := \mathtt{Hash}(\Gamma_0^j, ..., \Gamma_{2^h-1}^j)$ and sends it to $\mathcal{A}$.

4. Upon receiving $\chi, \chi_0, ..., \chi_{2^h-1} \in \mathtt{GR}(2^k, d)$ from $\mathcal{A}$, for $j \in [t]$, $S_{\mathcal{S}}$ computes $w_{0,\alpha_j}^j := M_j - g_0^j - \sum_{i\neq\alpha_j} v_{2i}^j$ and $w_{1,\alpha_j}^j := c_j - g_1^j - \sum_{i\neq\alpha_j} v_{2i+1}^j$. Next, $S_{\mathcal{S}}$

computes $W_j := \sum_{i \neq \alpha_j} \chi_i (v^j_{2i} + \chi \cdot v^j_{2i+1}) + \chi_{\alpha_j}(w^j_{0,\alpha_j} + \chi \cdot w^j_{1,\alpha_j})$. *Finally,* $S_{\mathcal{S}}$ *samples* $X \xleftarrow{\$} \mathtt{GR}(2^k, d)$, *and sends* $X, \{V_j := W_j + X \cdot \chi_{\alpha^j} \cdot y_j\}_{j \in [t]}$ *to* $\mathcal{A}$.

5. $S_{\mathcal{S}}$ *sets* $w^j_{0,i} := v^j_{2i}$, *for* $j \in [t], i \in [0, 2^h), i \neq \alpha_j$. $S_{\mathcal{S}}$ *sets* $\boldsymbol{e}^j \in \mathtt{GR}(2^k, d)^{2^h}$ *such that* $e^j_{\alpha_j} := y_j$, *and* $e_i = 0$ *for* $i \neq \alpha_j$. $S_{\mathcal{S}}$ *sends* $(\boldsymbol{w}^1_0, ..., \boldsymbol{w}^t_0)$, $(\boldsymbol{e}^1, ..., \boldsymbol{e}^t)$ *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* $(\boldsymbol{e}^1, ..., \boldsymbol{e}^t)$ *and compute* $(\boldsymbol{w}^1_0, ..., \boldsymbol{w}^t_0)$, *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* `Init` *phase. Later in the* `Extend` *phase,* $S_{\mathcal{R}}$ *records the input* $\boldsymbol{K}, \boldsymbol{a} \in \mathtt{GR}(2^k, d)^t$ *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* $\{K^j_{1,h+1}, \Gamma^j\}_{j \in [t]}$, *for* $l \in [0, 2^h)$, $S_{\mathcal{R}}$ *runs* $\mathtt{GGM.Eval}'(\alpha^l, \{K^j_{\bar{\alpha}^l_i, i}\}_{i \in [h]}, K^j_{1,h+1})$ *and gets* $(\{(v^{j,l}_{2i}, v^{j,l}_{2i+1})\}_{i \neq \alpha^l}, \{\Gamma^{j,l}_i\}_{i \in [0,2^h)})$. $S_{\mathcal{R}}$ *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* $I^j := \{l \in [0, 2^h) \mid \Gamma^j_l = \Gamma^j\} \subseteq [0, 2^h)$. *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* $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{mpVOLE}}$. *If it aborts, then* $S_{\mathcal{R}}$ *aborts.*

4. $S_{\mathcal{R}}$ *builds a set* $J := \{j \in [t] \mid |I_j| = 1\}$. *If there exists some* $l \neq l' \in I^j$, $j \in [t] \backslash J$ *and* $i \neq l, l'$ *such that* $v^{j,l}_{2i} \neq v^{j,l'}_{2i}$, $S_{\mathcal{R}}$ *aborts. Thus, for* $j \in [t] \backslash J$, $S_{\mathcal{R}}$ *can obtain consistent* $\{(v^j_{2i}, v^j_{2i+1})\}_{i \in [0,2^h)}$. *Besides, for* $j \in J$, *suppose* $I^j = \{\alpha^j\}$, *then* $S_{\mathcal{R}}$ *can simply set* $v^j_{2\alpha^j} = v^j_{2\alpha^j+1} = 0$ *to obtain* $\boldsymbol{v}^j_0, \boldsymbol{v}^j_1$.

5. Upon receiving $\{g^j_0, g^j_1\}_{j \in [t]}$ *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* $\beta^j_0 := g^j_0 - (K_j - \sum_{i \in [0,2^h)} v^j_{2i})$ *and* $\beta^j_1 := g^j_1 - (a_j - \sum_{i \in [0,2^h)} v^j_{2i+1})$. *If there exists a* $j \in [t]$ *such that* $\beta^j_1 \neq 0$, *but* $\beta^j_0 + \chi \cdot \beta^j_1 = 0$, $S_{\mathcal{R}}$ *aborts.* $S_{\mathcal{R}}$ *computes* $\hat{V}_j := \sum_{i \in [0,2^h)} \chi_i (v^j_{2i} + \chi \cdot v^j_{2i+1})$.

6. Upon receiving $\{V_j\}_{j \in [t]}$ *and* X *from* $\mathcal{A}$, *for* $j \in [t]$ *with* $\beta^j_1 \neq 0$, $S_{\mathcal{R}}$ *tries to find* $\hat{\alpha}^j \in I^j$ *such that* $\hat{V}_j - V_j = \chi_{\hat{\alpha}^j}(\beta^j_0 + \chi \cdot \beta^j_1)$. *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* $\beta^j_1 \neq 0$, $S_{\mathcal{R}}$ *resets* $I^j := \{\hat{\alpha}^j\}$, *and sends* I^j *to* $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{mpVOLE}}$. *If it aborts, then* $S_{\mathcal{R}}$ *aborts.*

8. $S_{\mathcal{R}}$ *sends* $(\boldsymbol{v}^1_0, ..., \boldsymbol{v}^t_0)$ *to* $\mathcal{F}^{\mathtt{GR}(2^k,d)}_{\mathrm{mpVOLE}}$.

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 $(v_{2i}^{j,l}, v_{2i+1}^{j,l}, \Gamma_i^{j,l}) = G'(\rho)$ and $(v_{2i}^{j,l'}, v_{2i+1}^{j,l'}, \Gamma_i^{j,l'}) = G'(\rho')$. Thus, we have that $\rho \neq \rho'$. By the right-half injectivity of G', we have $\Gamma_i^{j,l} \neq \Gamma_i^{j,l'}$, which leads to $\Gamma_l^j \neq \Gamma_{l'}^j$ overwhelmingly. However, $\Gamma_l^j = \Gamma_{l'}^j = \Gamma^j$ since $l, l' \in I^j$. This completes the proof of the above claim.

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

If $\mathcal{A}$ sends $(\hat{g}_0^j := g_0^j + \beta_0^j, \hat{g}_1^j := g_1^j + \beta_1^j)$ instead of correct (g_0^j, g_1^j), there will be a bias $\chi_{\alpha^j}(\beta_0^j + \chi \cdot \beta_1^j)$ for W_j computed by honest $P_{\mathcal{S}}$. Therefore, it can be observed that if $\mathcal{A}$ successfully guesses the α^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 α^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.