Efficient Information-Theoretic Secure Multiparty Computation over $\mathbb{Z}/p^k\mathbb{Z}$ via Galois Rings

Mark Abspoel[1,2(✉)], Ronald Cramer[1,3], Ivan Damgård[4], Daniel Escudero[4], and Chen Yuan[1]

[1] Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands
abspoel@cwi.nl
[2] Philips Research, Eindhoven, The Netherlands
[3] Mathematisch Instituut, Leiden University, Leiden, The Netherlands
[4] Aarhus University, Aarhus, Denmark

Abstract. At CRYPTO 2018, Cramer et al. introduced a secret-sharing based protocol called $\text{SPD}\mathbb{Z}_{2^k}$ that allows for secure multiparty computation (MPC) in the dishonest majority setting over the ring of integers modulo 2^k, thus solving a long-standing open question in MPC about secure computation over rings in this setting. In this paper we study this problem in the information-theoretic scenario. More specifically, we ask the following question: Can we obtain information-theoretic MPC protocols that work over rings with comparable efficiency to corresponding protocols over fields? We answer this question in the affirmative by presenting an efficient protocol for robust Secure Multiparty Computation over $\mathbb{Z}/p^k\mathbb{Z}$ (for *any* prime p and positive integer k) that is perfectly secure against active adversaries corrupting a fraction of at most 1/3 players, and a robust protocol that is statistically secure against an active adversary corrupting a fraction of at most 1/2 players.

1 Introduction

Secure Multiparty Computation (MPC) is a technique that allows several parties to compute any functionality in secret inputs, while revealing nothing more than the output, even if an adversary corrupts t of the n parties.

Several flavors of MPC exist, depending on the desired security level and threat model considered. A protocol is perfectly secure if an adversary's view of the protocol can be simulated given only his inputs and outputs, and where the simulated view follows exactly the same distribution as the real view. It is statistically secure if the statistical distance between the views is negligible in a security parameter. We will say that a protocol is information-theoretically secure if it is either perfectly or statistically secure.

MPC has been a very active area of research since the 1980s, beginning with the seminal work of Yao on garbled circuits. Since then, many theoretical and practical results have been found by the community, extending the knowledge about what is possible, and increasing efficiency. However, almost all the progress

D. Hofheinz and A. Rosen (Eds.): TCC 2019, LNCS 11891, pp. 471–501, 2019.
https://doi.org/10.1007/978-3-030-36030-6_19

has focused on arithmetic circuits over finite fields (even Boolean circuits are a special case of this). On the other hand, it is clearly also interesting to securely compute functions that are defined over other rings, such as $\mathbb{Z}/p^k\mathbb{Z}$, the ring of integers modulo p^k, where p is a prime and k is a positive integer. From a practical point of view, for instance, computing modulo 2^{32} or 2^{64} is close to what standard CPUs do. Closely matching the data format used by CPUs is an advantage since one expects that when programming secure computation one can reuse some of the techniques that CPUs use to run efficiently. Additionally, bitwise operations like comparison or bit decomposition are expressed more naturally modulo powers of 2, and are very fast when computed over these rings [1].

This observation has been confirmed in practice [13]. For example, for replicated secret sharing, protocols over rings like $\mathbb{Z}/2^{64}\mathbb{Z}$ can provide up to 8× savings in runtime and memory usage with respect to the field counterpart for some specific applications like neural network evaluation, which are heavy in terms of comparisons [2].

Thus, the natural question is: *can we design protocols that work directly over* $\mathbb{Z}/p^k\mathbb{Z}$ *and have efficiency close to what we can obtain for fields?*

This question was solved recently in the setting of dishonest majority (i.e. $t \leq n-1$), where cryptography is required to provide security, with the introduction of the SPD$\mathbb{Z}_{2^k}$ protocol [8]. This protocol computes with computational security circuits defined over the ring $\mathbb{Z}/2^k\mathbb{Z}$ (in fact, 2 can be replaced by any prime). This is achieved mainly by the introduction of information-theoretic MACs that work over rings with zero divisors and non-invertible elements, like $\mathbb{Z}/p^k\mathbb{Z}$. The efficiency is similar to the SPDZ and MASCOT protocols [12,14] which are state-of-the-art for dishonest majority MPC over finite fields.

However, the question has remained open for the case of honest majority where we can hope to get better (information-theoretic) security. It is also expected that in this setting the computational efficiency improves due to the fact that the computation needed for information-theoretically secure protocols tends to be simpler, as it is independent of a computational security parameter.

1.1 Our Contributions

In this work we resolve the above open question. Our solution relies on several key ingredients, which may be interesting in their own right. We give an overview below.

The first ingredient is a new secret sharing scheme that allows us to do "Shamir-style" sharing of elements in $\mathbb{Z}/p^k\mathbb{Z}$. We begin by noticing that Shamir secret sharing works over $\mathbb{Z}/p^k\mathbb{Z}$ as long as the secret is shared by at most $p-1$ players. In order to accomodate more players whilst maintaining a constant p, our key solution is to move to a Galois ring $R = (\mathbb{Z}/p^k\mathbb{Z}[x])/(f(x))$, where $f(x) \in (\mathbb{Z}/p^k\mathbb{Z})[x]$ is a monic polynomial of degree d such that $\overline{f}(x) \in \mathbb{F}_p[x]$, its reduction modulo p, is irreducible. We get a secret-sharing scheme over R using polynomial interpolation that works with $p^d - 1$ parties, so using the fact that R is a free module over $\mathbb{Z}/p^k\mathbb{Z}$ of rank d, we can embed $\mathbb{Z}/p^k\mathbb{Z}$ into the first coordinate of R and get an arithmetic secret-sharing scheme for $\mathbb{Z}/p^k\mathbb{Z}$.

Since we need that p^d is at least the number of players n, this incurs an overhead of $\log_p(n)$. To secret-share an element in $\mathbb{Z}/p^k\mathbb{Z}$ in this manner, each player gets an element in R as his share, which can be represented as $\log(n)$ elements in $\mathbb{Z}/p^k\mathbb{Z}$.

In terms of computational complexity, sharing an element requires $O(n^2\text{polylog}(n))$ ring operations which is an improvement over the black-box approach from [10]. It is known that the FFT-algorithms for operations over degree-d finite field extensions, as well as operations on polynomials over such fields, carry over to degree-d Galois rings, preserving quasi-linear (in d) computational complexity when working over our ring R [6].

For the remaining key ingredients, we distinguish between two models of MPC: perfectly secure MPC with $t < n/3$ assuming secure channels, and statistically secure MPC with $t < n/2$ in a setting where broadcast is given.

In the setting of perfectly secure MPC with $t < n/3$, we show that we can efficiently perform robust reconstruction in the presence of errors, we show that the hyperinvertible matrices needed in the protocol can be obtained over R can be obtained by lifting them from the residue field, and we show how to get MPC over $\mathbb{Z}/p^k\mathbb{Z}$ by efficient verification of the inputs, using techniques from [7]. We give the modifications needed to the protocol of [4], to obtain MPC over $\mathbb{Z}/p^k\mathbb{Z}$ with the communication complexity for a circuit C of size $|C|$ of $O(n\log(n)|C|)$ elements in $\mathbb{Z}/p^k\mathbb{Z}$.

For the setting where $t < n/2$ and broadcast is given, we develop a way to reduce the soundness error when checking whether values are secret-shared correctly.[1] We also show a packing technique that allows us to reduce the overhead to obtain a total communication complexity $O(|C|n^2\log n)$ ring elements, plus some term that does not depend on the size of the circuit. Finally, to get MPC over $\mathbb{Z}/p^k\mathbb{Z}$ rather than R, we show how to efficiently sample R-sharings of random elements of $\mathbb{Z}/p^k\mathbb{Z}$ with statistical security. These ideas allow us to adapt the protocol of Beerliova and Hirt [3]. We chose to adapt this protocol rather than the state-of-the-art of [5], because it allows for a simpler exposition of our novel techniques.

The protocols we get for the two settings are both a $\log(n)$ factor away from their original results due to the extension of $\mathbb{Z}/p^k\mathbb{Z}$ to R. Follow-up work by some of the authors provides a way to amortize away this factor, by using so-called "reverse multiplication-friendly embeddings" from algebraic geometric codes over rings with asymptotically good parameters [11].

1.2 Outline of the Document

In Sect. 2 we introduce the preliminaries for the rest of the work. This includes basic notation, Shamir secret-sharing over commutative rings, and the notion

[1] A problem that arises here is that the error probability of the protocol is not automatically negligible even if p^k is large. This is in contrast to the case of finite fields where the error probability usually is $1/|\mathbb{F}|$, where $|\mathbb{F}|$ is the order of the field. As we shall explain, by taking an extension of Galois rings $R \subset \hat{R}$ (where R is a subring), we can reduce the error.

of Galois rings. Then, in Sect. 3 we present our protocol for perfectly secure MPC over $\mathbb{Z}/p^k\mathbb{Z}$ with a corruption threshold of $t < n/3$. Section 4 discusses our protocol for statistically secure MPC over $\mathbb{Z}/p^k\mathbb{Z}$ in the honest majority setting. Finally, in Sect. 5 we present some conclusions and future work.

2 Preliminaries

2.1 Notation

$\mathbb{Z}$ denotes the ring of integers. For $m \in \mathbb{Z}$, $m\mathbb{Z}$ denotes the ideal $\{m \cdot n \mid n \in \mathbb{Z}\}$, and $\mathbb{Z}/m\mathbb{Z}$ denotes the quotient ring, which we regard as the ring of integers modulo m. For a ring R, let $R[X]$ denote the ring of polynomials in the variable X with coefficients in R. For an integer $m \geq 0$, let $R[X]_{\leq m} \subset R[X]$ denote the set of polynomials in $R[X]$ of degree at most m; it is an R-module. We denote by R^* the multiplicative subgroup of invertible elements in R.

2.2 Polynomial Interpolation over Commutative Rings

In this section, we will construct secret-sharing schemes over an arbitrary commutative ring. It will be the building block for the MPC protocols presented in this article. We begin by recalling some notions on polynomial interpolation, and on how it follows from the Chinese Remainder Theorem for rings; we follow the approach of Part II of [9].

Throughout this section, R will denote a commutative ring with multiplicative identity 1. Recall that an *ideal* of R is an additive subgroup $I \subseteq R$ such that $r \cdot x \in I$ for any $r \in R$, $x \in I$, i.e., an R-submodule. For $x \in R$, (x) denotes the ideal generated by x, i.e., $(x) := \{r \cdot x \mid r \in R\}$. Given two ideals I, I', their product is defined as the ideal II' given by finite sums of products xy with $x \in I$, $y \in I'$, and their sum $I + I'$ is defined as the ideal given by all elements of the from $x + y$, where $x \in I$, $y \in I'$.

Now we state the Chinese Reminder Theorem over rings.

Theorem 1. *Let $I_1, \ldots, I_m$ be m ideals of R that are pairwise co-maximal, i.e., for each pair I, I' we have $I + I' = R$. Then, the map*

$$\begin{aligned} R/(I_1 \cdots I_m) &\to \quad R/I_1 \times \cdots \times R/I_m \\ r \bmod I_1 \cdots I_m &\mapsto \big(r \bmod I_1, \ldots, r \bmod I_m\big) \end{aligned}$$

is a ring isomorphism.

We now recall some notions and results on polynomials over rings.

Theorem 2. *Let $g(X), h(X) \in R[X]$ be two polynomials, with $h(X)$* monic *(i.e., its leading coefficient is equal to 1). Then, there are two unique polynomials $q(X), r(X) \in R[X]$ such that*

- $g(X) = h(X)q(X) + r(X)$, *and*
- $\deg r(X) < \deg h(X)$.

Corollary 1. *We have the following:*

1. *For any monic* $h(X) \in R[X]$ *where* $\deg h(X) = d$*, we have an* R*-module isomorphism*
$$\begin{array}{rcl} R[X]_{\leq d-1} & \xrightarrow{\sim} & R[X]/(h(X)) \\ g(X) & \mapsto & g(X) \bmod (h(X)). \end{array}$$
2. *If* $h(X) = X - \alpha$ *for some* $\alpha \in R$*, then*
$$\begin{array}{rcl} R[X]/(X-\alpha) & \xrightarrow{\sim} & R \\ g(X) \bmod (X-\alpha) & \mapsto & g(\alpha) \end{array}$$
is an isomorphism of R*-modules.*

The above properties lead to the following result:

Theorem 3. *Let* $\alpha_1, \ldots, \alpha_m \in R$ *be such that* $\alpha_i - \alpha_j$ *is invertible for every pair of indices* $i \neq j$*. We then have that the map*
$$\begin{array}{rcl} R[X]_{\leq m-1} & \rightarrow & R \times \cdots \times R \\ f(X) & \mapsto & \big(f(\alpha_1), \ldots, f(\alpha_m)\big) \end{array}$$
is an R*-module isomorphism. Hence, for any* $x_1, \ldots, x_m \in R$*, there exists a unique interpolating polynomial of degree at most* $m-1$ *such that* $f(\alpha_i) = x_i$ *for each* i*.*

Proof. Let $h(X) := (X - \alpha_1) \cdot \ldots \cdot (X - \alpha_m)$. By Corollary 1, we have that the map $R[X]_{\leq m-1} \rightarrow R[X]/(h(X))$ given by $f(X) \mapsto f(X) \bmod (h(X))$ is an R-module isomorphism.

Notice that since $\alpha_i - \alpha_j$ is invertible for every $i \neq j$, we have that the ideals $(X - \alpha_i)$ and $(X - \alpha_j)$ are co-maximal for every $i \neq j$; thus by Theorem 1 we have that the map
$$\begin{array}{rcl} R[X]/(h(X)) & \rightarrow & R[X]/(X-\alpha_1) \times \cdots \times R[X]/(X-\alpha_m) \\ f(X) \bmod h(X) & \mapsto & \big(f(X) \bmod X-\alpha_1, \ldots, f(X) \bmod X-\alpha_m\big) \end{array}$$
is an R-module isomorphism.

Finally, again by Corollary 1, we have that the map $R[X]/(X - \alpha_i) \rightarrow R$ given by $f(X) \bmod X - \alpha_i \mapsto f(\alpha_i)$ is an isomorphism for every $i = 1, \ldots, m$. $\square$

The above theorem thus shows that polynomial interpolation extends from the field to the ring case, provided that the evaluation points are not only pairwise distinct, but that their pairwise differences are invertible.

Definition 1. *Let* $\alpha_1, \ldots, \alpha_n \in R$*. We say that these points form an* exceptional sequence *if for each pair of integers* $1 \leq i, j \leq n$ *with* $i \neq j$ *it holds that* $\alpha_i - \alpha_j \in R^*$*. We define the* Lenstra constant *of* R *to be the maximum length of an exceptional sequence in* R*.*

We the theory seen this far we can already define Shamir-secret sharing over an arbitrary ring R.

Construction 1 (Shamir-secret sharing over R). Let R be a finite ring, and let $\alpha_0, \ldots, \alpha_n \in R$ be an exceptional sequence. Let t be any positive integer such that $t \leq n$. We define the R-module of *share vectors* $C = \{(f(\alpha_0), \ldots, f(\alpha_n)) \mid f \in R[X]_{\leq t}\}$. To secret-share an element $x \in R$, pick a uniformly random share vector $\mathbf{x} \leftarrow \{(x_0, \ldots, x_n) \in C \mid x_0 = x\}$, and set the i-th share to be $f(\alpha_i)$ for $i = 1, \ldots, n$. If x is secret-shared with each player P_i having a share x_i, we denote the share vector $\mathbf{x}$ by $[x]$.

Note that the number of players the secret-sharing scheme admits is bounded by the Lenstra constant minus 1. Combining Construction 1 with Theorem 3 we have the following.

Proposition 1. *Construction 1 provides t-privacy and $(t+1)$-reconstruction.*

2.3 Galois Rings

We now restrict our attention to *Galois rings*, which are very well suited to our setting, since they contain $\mathbb{Z}/p^k\mathbb{Z}$ as a subring and have a relatively high Lenstra constant. For proofs of the assertions in this subsection, we refer the reader to [16].

Definition 2. *A* Galois ring *is a ring of the form $R := (\mathbb{Z}/p^k\mathbb{Z})[Y]/(h(Y))$, where p is a prime number, k is a positive integer, and $h(Y) \in (\mathbb{Z}/p^k\mathbb{Z})[Y]$ is a non-constant, monic polynomial such that its reduction modulo p is an irreducible polynomial in $\mathbb{F}_p[Y]$.*

Proposition 2. *Let R as in the above definition. It has the following properties:*

1. *R is a local ring, i.e. it has a unique maximal ideal $(p) \subsetneq R$. We have that $R/(p) \cong \mathbb{F}_{p^d}$, where d denotes the degree of h. In particular, we have a homomorphism $\pi : R \to \mathbb{F}_{p^d}$ that is "reduction modulo p".*
2. *The Lenstra constant of R is p^d.*
3. *For any prime p, positive integer k, and positive integer d there exists a Galois ring as defined above, and any two of them with identical parameters p, k, d are isomorphic. We may therefore write $R = GR(p^k, d)$.*
4. *If e is any positive integer, then R is a subring of $\hat{R} = GR(p^k, d \cdot e)$. There is a non-constant monic polynomial $\hat{h} \in R[X]$ that is irreducible modulo p, such that $\hat{R} = R[X]/(\hat{h}(X))$.*

Remark 1. Let $R = GR(p^k, d)$ be a Galois ring. Then there exists an $\mathbb{Z}/p^k\mathbb{Z}$-module isomorphism $\left(\mathbb{Z}/p^k\mathbb{Z}\right)^d \to R$, that sends each element $\mathbf{e}_j = (0, \ldots, 1, \ldots, 0)$ of the canonical basis of $\left(\mathbb{Z}/p^k\mathbb{Z}\right)^d$ to $Y^j \bmod (h(Y))$. Also, we have a natural ring embedding $\mathbb{Z}/p^k\mathbb{Z} \hookrightarrow R$, given by $x \mapsto x \bmod h(Y)$.

Moreover, there is another way to uniquely represent the elements of R. We have $R/(p) \cong \mathbb{F}_{p^d}$ and there exists a non-zero element $\xi \in R^*$ of multiplicative order $p^d - 1$. By defining the subset $\mathcal{I} = \{0, 1, \xi, \ldots, \xi^{p^d-2}\} \subset R$, it turns out that any element $a \in R$ can be uniquely written as $a = \sum_{i=0}^{k-1} a_i \cdot p^i$ where $a_0, \ldots, a_{k-1} \in \mathcal{I}$. Note that the homomorphism $\pi : R \to \mathbb{F}_{p^d}$ that is reduction modulo p from Item 1 in Proposition 2 is defined by $\pi(a) = a_0$.

This decomposition also allows us to define "division by powers of p". Indeed, notice that given an element $a = a_0 + a_1 p + a_2 p^2 + \cdots + a_{k-1}p^{k-1} \in R$ and a positive integer u, we have that p^u divides a if and only if $a_i = 0$ for all $i < u$. If this is the case, we then define $a/p^u := a_u + a_{u+1}p + \cdots + a_{k-1}p^{k-u-1}$; notice that $a/p^u \equiv a_u \pmod{p}$. If u is maximal and a is non-zero in R, then $a/p^u \in R^*$.

3 Perfectly Secure MPC for $t < n/3$ over Galois Rings

We assume that the computation is performed by n players, connected by a complete network of secure and authenticated channels. Let p be a prime number and k a positive integer; t players are under the control of a malicious, computationally unbounded adversary, where $t < n/3$. The adversary can be adaptive and rushing.

We adapt the protocol of [4], which uses three algebraic tools: the interpolation of a polynomial, hyper-invertible matrices and efficient error correction in Reed-Solomon codes. In the original protocol, these tools are defined over finite fields. In this section, we provide analogues of these tools over Galois rings. Note that the first tool, polynomial interpolation, is already given in Construction 1.

With these new tools, we obtain secure computation over any Galois ring R that has a Lenstra constant of at least $n + 1$. By taking the Galois ring to be large enough, we can accommodate any number of players. In Sect. 3.4, we show how to we obtain secure computation over $\mathbb{Z}/p^k\mathbb{Z}$ from computation over R. For passive security this is automatic, but for active security this requires verification of the inputs.

3.1 Hyper-Invertible Matrices

Hyper-invertible matrices are introduced in [4] to efficiently obtain secret-shared randomness in MPC protocols with active security. Here we summarize their definition and fundamental properties, generalized to hold over rings.

Definition 3. *A matrix* $\mathbf{M} \in R^{u \times u'}$ *is* hyper-invertible *if for any row index set* $I \subseteq \{1, \ldots, u\}$ *and column index set* $J \subseteq \{1, \ldots, u'\}$ *with* $|I| = |J| > 0$*, the matrix* $\mathbf{M}_I^J$ *is invertible, where* $\mathbf{M}_I$ *denotes the submatrix of* $\mathbf{M}$ *with rows in* I*,* $\mathbf{M}^J$ *denotes the submatrix of* $\mathbf{M}$ *with columns in* J*, and* $\mathbf{M}_I^J := (\mathbf{M}_I)^J$*.*

Construction 2. Let n and k be positive integers, and let p be a prime number. Further, let $R = GR(p^k, d)$ with $p^d \geq 2n$, and let $\alpha_1, \ldots, \alpha_{2n}$ be an exceptional sequence in R. Applying Theorem 3 twice, we get an R-module isomorphism from R^n to R^n, sending $(f(\alpha_1), \ldots, f(\alpha_n)) \mapsto (f(\alpha_{n+1}), f(\alpha_{n+2}), \ldots, f(\alpha_{2n}))$. It is represented by an $n \times n$ matrix over R which is hyper-invertible. The proof of this fact follows the lines of its analogous proof over fields, and we refer the reader to [4] for details.

Hyper-invertible matrices have the following key property. The proof from [4] carries over, given the properties that we have shown for R.

Lemma 1. *Let* $\mathbf{M} \in R^{n\times n}$ *be an* n*-by-*n *hyper-invertible matrix, and let* $I, J \subseteq \{1,\ldots,n\}$ *be index sets such that* $|I| + |J| = n$*. Then, there exists a linear isomorphism* $\varphi = \varphi_{I,J} : R^n \to R^n$ *such that for any* $\mathbf{x}, \mathbf{y} \in R^n$ *it holds that* $\varphi(\mathbf{x}_J, \mathbf{y}_I) = (\mathbf{x}_{\bar{J}}, \mathbf{y}_{\bar{I}})$*, where* $\bar{R}$ *and* $\bar{C}$ *denote the complements* $\{1,\ldots,n\}\setminus R$ *and* $\{1,\ldots,n\}\setminus C$*, respectively.*

3.2 Robust Reconstruction

Recall from Construction 1 we have an R-module $C = \{(f(\alpha_1),\ldots,f(\alpha_n)) \mid f \in R[X]_{\leq t}\}$ of share vectors. We wish to have *robust reconstruction*: a party P that receives shares x_i for $i = 1,\ldots,n$, where $x_i = f(\alpha_i)$ for "most" values of i, should be able to reconstruct the correct secret $f(\alpha_0)$ even some shares are corrupted, e.g., they contain arbitrary elements of R.

This is also known as the *decoding problem* of linear codes. When R is a finite field, R-vector spaces of the form C as above are known as (generalized) *Reed-Solomon codes*. We want an algorithm that does the following. As input we give a vector $(x_1,\ldots,x_n) \in (R \cup \{\perp\})^n$ such that there exists some $f \in R[X]_{\leq t}$ with $x_i = f(\alpha_i)$ for all $i = 1,\ldots,n$ except for at most $\lfloor\frac{n-t-1}{2}\rfloor$ positions. As output, the algorithm has to produce f.

We assume black-box access to a decoding algorithm for Reed-Solomon codes (i.e. for vector spaces of the form C as above when R is a finite field), such as the Berlekamp-Massey algorithm [15]. We show how to obtain a decoding algorithm for C over a Galois ring $R = GR(p^k, d)$ that makes k calls to the algorithm over fields.

We fix an exceptional sequence $\alpha_1,\ldots,\alpha_n \in R$. Recall from Remark 1 that any element $a \in R$ can be uniquely written as

$$a = a_0 + a_1 p + a_2 p^2 + \cdots + a_{k-1}p^{k-1}$$

where $a_0,\ldots,a_{k-1} \in \mathcal{I} = \{0, 1, \xi, \ldots, \xi^{p^d-2}\}$. It follows that for $f(X) \in R[X]_{\leq t}$, we can uniquely write $f(X)$ as $f(X) = f_0(X) + pf_1(X) + \cdots + p^{k-1}f_{k-1}(X)$, where $f_0(X),\ldots,f_{k-1}(X) \in \mathcal{I}[X]_{\leq t}$. Moreover, we have

$$f(\alpha_i) \equiv \sum_{i=0}^{j-1} p^i f_i(\alpha_i) \pmod{p^j}.$$

Since $\alpha_1,\ldots,\alpha_n$ have their pairwise differences invertible, this means they map to distinct elements modulo p. For each $i = 1,\ldots,n$ let $\beta_i = \pi(\alpha_i) \in \mathbb{F}_{p^d}$ where $\pi : R \to \mathbb{F}_{p^d}$ is the reduction modulo p from Item 1 of Proposition 2. Notice that π gives this one-to-one correspondence between $\mathcal{I}$ and $\mathbb{F}_{p^d}$. In particular, the inverse π^{-1} is a well-defined function onto $\mathcal{I}$.

Theorem 4. *The protocol of Fig. 1 can correct up to* $\lfloor\frac{n-t-1}{2}\rfloor$ *errors with* k *calls to the decoding algorithm over* $\mathbb{F}_{p^d}$*.*

Decoding Reed-Solomon Codes over a Galois Ring R

- Input: $\mathbf{x} = (x_1, \ldots, x_n) \in (R \cup \{\perp\})^n$.
- Let $\mathbf{y} \leftarrow \mathbf{x}$. For $i = 0, \ldots, k-1$ perform the following operations:
  1. $\mathbf{y} \leftarrow \pi(\mathbf{y}/p^i)$, applied element-wise.
  2. Run the decoding algorithm the input $\mathbf{y}$ and let the $\bar{f}_i(X)$ be the output polynomial. Let $f_i(X) = \pi^{-1}(\bar{f}_i(X)) \in \mathcal{I}[X]_{\leq t}$.
  3. Let $t_j = \sum_{\ell=0}^{i} p^\ell f_\ell(\alpha_j)$ for $j = 1, \ldots, n$ and $\mathbf{y} \leftarrow (x_1 - t_1, \ldots, x_n - t_n)$.
  4. If there exists j such that $x_j - t_j$ is not divisible by p^{i+1}, we claim an error in index j and set $\perp$ on the j-th component of $\mathbf{y}$.
- Output: $f(X) = f_0(X) + pf_1(X) + \cdots + p^{k-1}f_{k-1}(X)$.

Fig. 1. Decoding Reed-Solomon Codes over a Galois Ring R

Proof. Let us justify this decoding algorithm. We start with $i = 0$. Note that $\bar{f}_0(X) = \pi(f_0(X)) \in \mathbb{F}_{p^d}[X]_{\leq t}$. Thus,

$$\mathbf{c}_f = (\bar{f}_0(\beta_1), \ldots, \bar{f}_0(\beta_n))$$

is a vector in the corresponding Reed-Solomon code over $\mathbb{F}_{p^d}$. Since $\mathbf{y} = \pi(\mathbf{x})$ is a corrupted vector in $\mathbb{F}_{p^d}^n$ differing in at most $\lfloor \frac{n-t-1}{2} \rfloor$ positions from $\mathbf{c}_f$, the decoding algorithm over $\mathbb{F}_{p^d}$ will recover $\bar{f}_0(X)$ and then $f_0(X)$. Now, assume that we have already recovered $f_0(X), \ldots, f_i(X)$. Let us fix x_j, the j-th component of $\mathbf{x}$. Assume that x_j is not corrupted, i.e., $x_j = f(\alpha_j)$. Then, we have

$$x_j - t_j = f(\alpha_j) - \sum_{\ell=0}^{i} p^\ell f_\ell(\alpha_j) = p^{i+1} \sum_{\ell=0}^{k-i-2} p^\ell f_{\ell+i+1}(\alpha_j).$$

This implies $x_j - t_j$ is divisible by p^{i+1}. Moreover, $\pi((x_j - t_j)/p^{i+1}) = \pi((f_{i+1}(\alpha_j)) = \bar{f}_{i+1}(\beta_j)$. Thus $\pi(\mathbf{y}/p^i)$ agrees with $(\bar{f}_{i+1}(\beta_1), \ldots, \bar{f}_{i+1}(\beta_N))$ in the position that is not corrupted. It follows that $\pi(\mathbf{y}/p^i)$ differs in at most $\lfloor \frac{n-t-1}{2} \rfloor$ positions from $(\bar{f}_{i+1}(\beta_1), \ldots, \bar{f}_{i+1}(\beta_N))$. Running the decoding algorithm over $\mathbb{F}_{p^d}$on $\pi(\mathbf{y}/p^i)$ will output the polynomial $f_{i+1}(X)$. The desired result follows as we only invoke the decoding algorithm over the finite field k times. □

3.3 MPC over R

Let d be the smallest positive integer with $p^d \geq 2n$, and write $R = GR(p^k, d)$. Let $(\alpha_0, \alpha_1, \ldots, \alpha_n)$ and $(\beta_1, \ldots, \beta_{2n})$ be exceptional sequences of R of respective lengths $n+1$ and $2n$.

We replace some of the components of [4] to extend this protocol over rings. We use the n-player Shamir-like secret-sharing scheme obtained in Construction 1, where α_i is assigned to each player P_i. Thus both the share and secret lie in R. Also, we use the hyper-invertible matrices from Construction 2, with evaluation points $\beta_1, \ldots, \beta_{2n}$; and we recover secrets from $n' \leq n$ shares with t' corruptions, provided that $t < n' - 2t'$, using the procedure in Fig. 1.

With these tools in place, the remainder of the protocol from [4] can be used to obtain MPC over R, as encapsulated in the following theorem.

Theorem 5. *There exists an efficient MPC protocol over the Galois Ring $R = GR(p^k, d)$ with $p^d \geq 2n$, for n parties, that is secure against the maximal number of active corruptions $\lfloor (n-1)/3 \rfloor$, and that has an amortized communication complexity of $O(n)$ ring elements per gate.*

3.4 MPC over $\mathbb{Z}/p^k\mathbb{Z}$

From Theorem 5, we get MPC over $R = GR(p^k, d)$ with $p^d \geq 2n$, but this does not give us MPC over $\mathbb{Z}/p^k\mathbb{Z}$ for an arbitrary number of players. We can embed inputs in $\mathbb{Z}/p^k\mathbb{Z}$ into R, but we do need to verify that the original inputs are actually in $\mathbb{Z}/p^k\mathbb{Z}$.

Proving that a secret-shared value $[a]$ is in $\mathbb{Z}/p^k\mathbb{Z}$ reduces to sampling a secret-shared random element $[r] \leftarrow \mathbb{Z}/p^k\mathbb{Z}$, as follows: to check that $a \in \mathbb{Z}/p^k\mathbb{Z}$ we simply locally compute $[a + r]$ and open the result. We have that $a \in \mathbb{Z}/p^k\mathbb{Z}$ if and only if $a + r \in \mathbb{Z}/p^k\mathbb{Z}$. Also, since r is a uniformly random element in $\mathbb{Z}/p^k\mathbb{Z}$, $a + r$ does not reveal any information about a (if a is in fact in $\mathbb{Z}/p^k\mathbb{Z}$).

We use an idea from [7] to generate these sharings of random elements in $\mathbb{Z}/p^k\mathbb{Z}$. Since R is a free module over $\mathbb{Z}/p^k\mathbb{Z}$ of rank d, we can write down a basis of R. In fact, a power basis $1, \xi, \ldots, \xi^{d-1}$ exists. After fixing ξ, an element $b \in R$ can thus be uniquely written $b = b_0 + b_1\xi + \cdots + b_{d-1}\xi^{d-1}$, and we can identify b with its coefficient vector $(b_0, \ldots, b_{d-1})$. The map $\phi : R \to (\mathbb{Z}/p^k\mathbb{Z})^d$ such that $\phi(b) = (b_0, \ldots, b_{d-1})$ is a $\mathbb{Z}/p^k\mathbb{Z}$-module isomorphism.

Let $\lambda \in R$. Multiplication by λ in R defines an R-module endomorphism $R \to R$, which is in particular an $\mathbb{Z}/p^k\mathbb{Z}$-module homomorphism $(\mathbb{Z}/p^k\mathbb{Z})^d \to (\mathbb{Z}/p^k\mathbb{Z})^d$. Thus, this operation can be seen be represented as a $d \times d$ matrix M_λ with entries in $\mathbb{Z}/p^k\mathbb{Z}$ such that for any $b \in R$ it holds that $\phi(\lambda b) = M_\lambda \phi(b)$. This is similar to how elements in a field extension can be seen as matrices over the base field.

Now, let A be an $n \times n$ matrix with entries in R, for arbitrary $n \geq 1$, and let $(x_1, \ldots, x_n) \in R^n$ be a vector. Each entry x_i can in turn be represented as a vector $(x_{i,1}, \ldots, x_{i,d})$ with entries in $\mathbb{Z}/p^k\mathbb{Z}$ such that $x_i = \phi\left((x_{i,1}, \ldots, x_{i,d})\right)$. The action of A on R^n is R-linear so in particular $\mathbb{Z}/p^k\mathbb{Z}$-linear. If we let $(y_1, \ldots, y_n)^T = A(x_1, \ldots, x_n)^T$ then each entry y_i is the R-linear combination $y_i = a_{i,1}x_1 + \cdots + a_{i,n}x_n$, where $(a_{i,1}, \ldots, a_{i,n}) \in R^n$ is the i-th row of A. Applying ϕ^{-1} to this equation we see that the $\mathbb{Z}/p^k\mathbb{Z}$-linear action of A on the elements $x_{i,j}$ is as follows

$$(y_{i,1}, \ldots, y_{i,d})^T = M_{a_{i,1}}(x_{1,1}, \ldots, x_{1,d})^T + \cdots + M_{a_{i,n}}(x_{n,1}, \ldots, x_{n,d})^T.$$

In Fig. 2, we present a protocol for constructing sharings over R of random elements in $\mathbb{Z}/p^k\mathbb{Z}$. The function of $\mathsf{RandEl}(\mathbb{Z}/p^k\mathbb{Z})$ is to amortize away the cost of generating sharings of random elements in $\mathbb{Z}/p^k\mathbb{Z}$ and meanwhile to verify if the shares correspond to a random element in $\mathbb{Z}/p^k\mathbb{Z}$ instead of R. Our protocol is similar to $\mathsf{RandElSub}(V)$ in [7]. Using player elimination, we assume that

RandEl$(\mathbb{Z}/p^k\mathbb{Z})$

Fixed public parameters: $1 \le T \le n'-2t'$, M an $n' \times n'$ hyper-invertible matrix over R given in Construction 2.

Processing: 1. For $i = 1, \dots, n'$, P_i selects d uniformly random elements $s_{i,1}, \dots, s_{i,d} \in \mathbb{Z}/p^k\mathbb{Z}$ and secret-shares each of them in parallel using the secret-sharing scheme in Construction 1 over R with n' players and t'-privacy. This can be interpreted as each party secret-sharing a vector of d elements, and we write $[\![\mathbf{s}_i]\!] := ([s_{i,1}], \dots, [s_{i,d}])$. This constitutes a secret-sharing where the *correct* secrets are elements of $(\mathbb{Z}/p^k\mathbb{Z})^d$ and the shares are elements of R^d.

2. Players locally compute $([\![\mathbf{r}_1]\!], \dots, [\![\mathbf{r}_{n'}]\!]) = M([\![\mathbf{s}_1]\!], \dots, [\![\mathbf{s}_{n'}]\!])$. Note that the matrix M is defined over R; the action on the individual R-sharings is defined via the matrices $M_{m_{i,j}}$ where $M = (m_{i,j})$.

3. For $i = T+1, \dots, n'$, every party P_j sends its share of $[\![\mathbf{r}_i]\!]$ to P_i. P_i then verifies the values received if the secret is indeed a vector in $(\mathbb{Z}/p^k\mathbb{Z})^d$, and if not, gets unhappy.

Output: If all honest players are happy, the $d \cdot T$ sharings $[r_{1,1}], \dots, [r_{1,d}], [r_{2,1}], \dots, [r_{T,d}]$ are sharings over R with each secret an independent uniformly random element from $\mathbb{Z}/p^k\mathbb{Z}$.

Fig. 2. Protocol for Generating Sharings of Random Elements in Subring

there are currently n' parties taking part in the computation (labeled $P_1, \dots, P_{n'}$ without loss of generality) and at most t' of them are corrupted. Note that $t < n' - 2t'$. If a party is unhappy, player elimination ensures that we can find a pair of players that contains at least one corrupted player. Like Proposition 4 in [7], we only need to communicate $O(n)$ elements in R per sharing of a random element in $\mathbb{Z}/p^k\mathbb{Z}$.

Proposition 3. *If all honest players are happy after the execution of* *RandEl*$(\mathbb{Z}/p^k\mathbb{Z})$*, then the output is correct, i.e. the* $d \cdot T$ *sharings* $[r_{1,1}], \dots, [r_{1,d}], [r_{2,1}], \dots, [r_{T,d}]$ *are correct sharings of uniformly random elements in* $\mathbb{Z}/p^k\mathbb{Z}$*, and the adversary has no information about these values, other than the fact that they belong to* $\mathbb{Z}/p^k\mathbb{Z}$*.*

With the help of Proposition 3 and our above analysis, we obtain the following theorem.

Theorem 6. *There exists an efficient* n*-party MPC protocol for circuits defined over* $\mathbb{Z}/p^k\mathbb{Z}$*, that is secure against the maximal number of active corruptions* $\lfloor (n-1)/3 \rfloor$*, and that has an amortized communication complexity of* $O(n \log n)$ *ring elements per gate.*

4 Statistically Secure MPC for Honest Majority over Galois Rings

In this section we present a protocol for secure computation over the Galois ring $R = GR(p^k, d)$ that is statistically secure against active adversaries. The protocol tolerates a number of corrupted parties $t < n/2$, which is optimal in this setting.

Our protocol is largely based on the dispute control protocol from [3]. However, some of their techniques explicitly use properties about fields, which do not apply to our setting directly. In this section we show that, due to some special properties of the Galois ring R (mostly the fact that R is local), most of these techniques actually apply to this setting as well, at the expense of having a higher failure probability than in the field case. More explicitly, when working over a field $\mathbb{F}$ it can be shown that the failure probability is roughly $1/|\mathbb{F}|$, but in our setting this probability is close to $1/p^d$, which is potentially far from $1/|R| = 1/p^{k \cdot d}$. In particular, this implies that d must be as large as the security parameter κ.

However, if we have our computation over $R = GR(p^k, d)$ with $p^d \geq n + 1$, so that we have enough interpolation points for each player, we can avoid much of the overhead. We do this by moving to an extension Galois ring $\hat{R} = GR(p^k, d \cdot \hat{d}) \supset R$ (see Proposition 2). For many subprotocols where the error depends on p^d, we can pack $\hat{d}$ values of R into $\hat{R}$ (since $\hat{R} \cong R^{\hat{d}}$ as R-modules), and keep the same amortized complexity. In particular, we do not get a total complexity that is linear in both the size of the circuit *and* the security parameter κ, which is what one would get if d were as large as κ.

To get computation over $\mathbb{Z}/p^k\mathbb{Z}$ where $p \leq n$, we embed $\mathbb{Z}/p^k\mathbb{Z} \hookrightarrow R$, but we do need to verify that the inputs are actually in $\mathbb{Z}/p^k\mathbb{Z}$, like we saw in Sect. 3.3. We will develop the machinery needed for this in Sect. 4.7.

4.1 Overview of Our Techniques

We begin by presenting a summary of the main novel techniques used to achieve the results in this section. The details of these, and their specific usage in the context of our protocol, are explained thoroughly in subsequent sections.

Error Checking. To guarantee correctness of the computation, we need a process that checks whether values are secret-shared correctly, with negligible error. Suppose we have secret-shared values $[x_1], \dots, [x_\ell]$ and we want to check whether the players have consistent shares, i.e. each reconstructing set of honest players jointly have shares that reconstruct to the same secret value. A trick commonly used over fields is to fix a random linear combination $y = r_1 x_1 + \cdots + r_\ell x_\ell$, for publicly known uniformly random values $r_1, \dots, r_\ell$, and to have the players broadcast the shares of y. They can then check whether their shares are consistent, e.g. for Shamir's secret-sharing scheme they check whether the shares are on a polynomial of degree of at most t.

This approach works over a finite field $\mathbb{F}$ since the inner product of any non-zero vector (an "error vector") with a uniformly random vector is zero with probability $1/|\mathbb{F}|$. Therefore any inconsistency in some value x_i is very likely to give an inconsistency in y. In other rings, this does not necessarily apply, and the product of a non-zero value times a random value is not necessarily random: for example, in $\mathbb{Z}/2^k\mathbb{Z}$ we have $\Pr[r \cdot 2^{k-1} = 0] = 1/2$ for uniformly random r.

For the Galois ring R, it turns out the above procedure does work, but only with error probability p^{-d}, i.e. it only scales in the *degree* of the Galois ring, not in its order $p^{k \cdot d}$. We illustrate this with the following protocol.

Consider the setting where we have a single dealer that secret-shares a single secret value $[x] \in R$ and a single verifier that wants to check whether $[x]$ is secret-shared correctly. To ensure privacy towards the verifier, the dealer also secret-shares a random value $[u] \in R$. The protocol runs as follows:

1. The dealer samples $u \in R$ and secret-shares $[x], [u]$ among the players.
2. The verifier samples $r \in R$ and broadcasts it to all players.
3. All players reconstruct $y = rx + u$ towards the verifier.
4. The verifier accepts if all received shares of y are consistent, and rejects otherwise.

This protocol is private because u is chosen uniformly random by the dealer. We shall now analyze the soundness error. It is useful to take a more general view, and let $C \subseteq R^n$ denote the set of vectors of consistent shares; recall C from Construction 1. More generally, let C be any free R-module, i.e. it has a basis. Note that the verifier accepts if $y \in C$, and the dealer cheats successfully if the verifier accepts and $x \notin C$.

We analyze the soundness error using a fact about roots of polynomials over R:

Lemma 2. *Let $f \in R[X]$ be a polynomial of arbitrary degree $\ell > 0$. Then $\Pr_{x \leftarrow R}[f(x) = 0] \leq \ell/p^d$, where x is drawn uniformly from R.*

Proof. Write $f(X) = a_0 + a_1 X + \cdots + a_\ell X^\ell$. Let u be the highest power of p such that p^u divides each coefficient $a_0, \ldots, a_\ell$ of f. Then, $f(X)/p^u$ has at least one coefficient invertible, hence its reduction $g := \overline{f(X)/p^u}$ modulo p is a nonzero polynomial of degree $\leq \ell$ over the field $R/(p)$ of order p^d. Clearly, if $f(x) = 0$ then $\overline{x} := x \bmod (p)$ is a root of g. Since g has at most ℓ roots, $g(\overline{x}) = 0$ with probability $\leq \ell/p^d$ for uniformly random $\overline{x}$. Since reduction modulo (p) is a homomorphism, in particular it has pre-images of equal size, hence given that x is uniformly random in R, $\overline{x}$ is uniformly random in $R/(p)$. □

Lemma 3. *Let $C \subseteq R^n$ be a free R-module. For all $x \notin C$ and $u \in R^n$, we have that*

$$\Pr_{r \leftarrow R}[rx + u \in C] \leq 1/p^d,$$

where r is chosen uniformly at random from R.

Proof. Let $g : R^n \to R$ be an R-module homomorphism such that $g(c) = 0$ for all $c \in C$, and such that $g(x) \neq 0$. Such a homomorphism in particular exists because C is free, and it is therefore a direct summand of R^n.

If $rx + u \in C$, then $0 = g(rx + u) = rg(x) + g(u)$, so r is a root of the linear polynomial $g(x)X + u$, which by the previous lemma occurs with probability $\leq 1/p^d$. □

Packing. To get a negligible correctness error for MPC over R, our solution is to move from R to an extension $R \subset \hat{R}$, where $\hat{R} = GR(p^k, d \cdot \hat{d})$ for an integer $\hat{d} > 1$ with $p^{d \cdot \hat{d}} \geq 2^\kappa$. However, the efficiency is unfavorable since communication and computation is $\Omega(\kappa n^2)$ per multiplication gate.

To improve efficiency, we observe that $\hat{R}$ is a free R-module of rank $\hat{d}$, i.e. $\hat{R} \cong R^{\hat{d}}$. Therefore, we can interpret an element of $\hat{R}$ as a vector of elements of R of length $\hat{d}$. This allows us to check $\hat{d}$ elements of R in parallel, by checking one element of $\hat{R}$. In $\hat{R}$ our correctness check has error probability $p^{-d \cdot \hat{d}} \leq 2^{-\kappa}$, and thus by moving to the extension we can both achieve the desired soundness error while getting no amortized overhead.

Let $g(Y)$ be a monic polynomial over R of degree $\hat{d}$ which is irreducible when taken modulo p, and let $\hat{R} = R[Y]/(g(Y))$. Let $w_1, \ldots, w_{\hat{d}}$ be a basis of $\hat{R}$ over R as a module and consider the natural isomorphism of modules $\psi : R^{\hat{d}} \to \hat{R}$ given by $\psi(x_1, \ldots, x_{\hat{d}}) = \sum_{i=1}^{\hat{d}} x_i \cdot w_i$.

Finally, consider $y \in \hat{R}$ with $\psi(y_1, \ldots, y_{\hat{d}}) = y$ and assume that y is secret-shared via a polynomial $F \in \hat{R}[X]$ and that the exceptional sequence $\alpha_1, \ldots, \alpha_n$ of evaluation points is in R. This polynomial can be written uniquely as $F(X) = \sum_{i=1}^{m} f_i(X) \cdot w_i$ where f_i are polynomials in $R[X]$. Moreover, we notice that for all $r \in R$ it holds that $F(r) = \psi(f_1(r), \ldots, f_{\hat{d}}(r))$, so in particular the polynomial f_i defines shares of y_i, for $i = 1, \ldots, \hat{d}$. Conversely, if we have shares of $y_1, \ldots, y_{\hat{d}}$ using polynomials $f_1, \ldots, f_{\hat{d}}$ over R, then we can define a share of $\psi(y_1, \ldots, y_{\hat{d}})$ over $\hat{R}$ which is given by the polynomial $F = \sum_{i=1}^{\hat{d}} f_i \cdot w_i$. We abuse notation and write $\psi([y]_{\hat{R}}) = ([y_1]_R, \ldots, [y_m]_R)$ to denote the situation above.

We then have the following:

Lemma 4. *Let $y \in \hat{R}$ and $(y_1, \ldots, y_m) = \psi^{-1}(y)$, and suppose that $\psi([y]_{\hat{R}}) = ([y_1]_R, \ldots, [y_m]_R)$. Then $[y]_{\hat{R}}$ is correctly shared if and only if each $[y_i]_R$ is correctly shared.*

Proof. Let F be the polynomial over $\hat{R}$ interpolating y and let f_i be the polynomial over R interpolating y_i, for $i = 1, \ldots, m$. We know that $F = \sum_{i=1}^{m} f_i \cdot w_i$, and since $w_1, \ldots, w_i$ is a basis for $\hat{R}$ over R it follows that $\deg(F) = \max\{\deg(f_1), \ldots, \deg(f_m)\}$. Therefore, in particular $\deg(F) \leq t$ if and only if $\deg(f_i) \leq t$ for all i. The desired result follows. □

MPC over $\mathbb{Z}/p^k\mathbb{Z}$. Like in Sect. 3.4, checking the membership of a secret-shared value in a Galois subring $S \subset R$ can be reduced to sampling a random secret-shared $[s]$, where $s \leftarrow S$ and the secret-sharing is over R. To check whether an input $[x]$ is in S, we can simply mask and open $x + s$, and check whether it is in S. This holds for any $x \in S$, since S is additively closed.

To get a random sharing $[s]$, a straightforward solution is to let each player P_i sample a random element s_i and secret-share it (over R). The players then compute $[s] = \sum_{i=1}^{n}[s_i]$. We can check the correctness of $[s]$ by using the method of Sect. 4.1, where we check a batch of many different values at once. However, in this situation, we are only allowed to take S-linear combinations. In particular, for $S = \mathbb{Z}/p^k\mathbb{Z}$, Lemma 3 only gives an error probability of $1/p$.

To reduce the error probability, we do the following. Let C be the set of share vectors $[s] = (s_1, \ldots, s_n)$ of secrets $s \in S$, with shares $s_1, \ldots, s_n \in R$. Note that C is an S-module but not an R-module in general. Since R is a free module over S, we have $R \cong S^e$ where $e = \operatorname{rank} R$. We may now take the extension of scalars of C to R via the following tensor product of S-modules:

$$\hat{C} := C \otimes_S R \cong C \otimes_S S^e$$

In contrast to C, we have that $\hat{C}$ *is* an R-module, and in fact an R-submodule of $R^n \otimes_S R \cong R^{n \cdot e}$. A dealer will secret-share a vector of e random elements of S in parallel over R. Each player thus obtains a vector of shares (with each entry in R), which can be interpreted as one element of $R \otimes_S R \cong R^e$. All of the players' shares together form a vector in $R^{n \cdot e}$, which is in $\hat{C}$ if indeed the e secret-shared elements are in S. We can now apply the methods from Sect. 4.1 to batch check these values with error probability $1/p^d$.

4.2 Computation over Fields

As a base for our protocol for statistically secure MPC in the honest majority setting, we choose the protocol from [3]. It maintains the invariant that every wire of the circuit is secret-shared using Shamir's secret-sharing scheme. Linear gates are given for free by the secret-sharing scheme, and multiplication gates are handled by means of some preprocessed data known as multiplication triples, which are generated themselves using a technique known as resharing. The protocol follows the traditional offline/online paradigm where the multiplication triples are generated during the so-called offline phase that is independent from the inputs, and these triples are subsequently used in the online phase to perform the actual secure computation.

With the secret-sharing scheme over rings from Sect. 2.2, adapting the basic resharing based protocol to the ring setting is straight-forward. Therefore, an efficient protocol for statistically secure computation with honest majority *with abort* and *over rings* can be easily developed at this point. However, in this work we aim for full security, and in order to provide robustness we need to adapt the tools introduced by the dispute control technique, and this becomes much more involved since these highly exploit the fact that the underlying structure is a field.

While we do not have the nice structural properties of fields, we are able to exploit properties of Galois rings to obtain sub-protocols with comparable efficiency to those over fields. In the rest of the section we will focus only on the algebraic aspects of dispute control that must be modified in order to adapt them to work over Galois rings. A second part of dispute control uses more "combinatorial" arguments which are independent of the underlying algebraic structure and therefore they apply directly to our setting. In these cases we refer the reader to the appropriate references.

4.3 Dispute Control

Dispute control is a technique used in [3] in order to provide guaranteed output delivery. Here the parties keep track of a publicly known dispute set Δ of unordered pairs $\{P_i, P_j\}$ of parties that are in dispute. We write $P_i \not\leftrightarrow P_j$ if $\{P_i, P_j\} \in \Delta$, and $P_i \leftrightarrow P_j$ otherwise. At a very high level, a new dispute $P_i \not\leftrightarrow P_j$ is generated whenever P_i thinks that P_j has cheated, or vice versa, and the protocol will guarantee that whenever a new dispute is generated then at least one of the two parties involved is corrupt (i.e. an honest party will never go in dispute with another honest party).

We let Δ_i denote the set of parties P_j such that $P_i \not\leftrightarrow P_j$. Let $\mathcal{X} \subseteq \mathcal{P}$ denote the set of parties P_i that have $|\Delta_i| > t$, i.e. parties that have a dispute with more than t other parties. They are universally known as corrupt, because no honest party can have a dispute with more than t other parties.

At a very high level, the way in which dispute control is used in the protocol is the following. The computation is divided into segments such that at the end of each segment there is a consistency check. If the check fails, the parties run a dispute control protocol that results in a new pair of players that are not yet in dispute, such that one of them is guaranteed to be corrupt.

Once the dispute has been identified, the segment is re-run. There can be at most $t(t+1)$ disputes. By dividing the computation into n^2 segments of approximately equal length, the overhead of repeating failed segments is at most a factor of 2. In this work we will not focus on the details of dispute control and we only introduce it as we will need the notation. For the details of dispute control see [3].

4.4 1D, 2D and 2D* Sharings

As before, let $h(Y) \in (\mathbb{Z}/p^k\mathbb{Z})[Y]$ be a monic polynomial of degree d such that its reduction mod p is irreducible, and let R be the Galois ring $(\mathbb{Z}/p^k\mathbb{Z})[Y]/(h(Y))$. We assume that $d \geq \log_p(1+n)$, so that there is an exceptional sequence $0, \alpha_1, \ldots, \alpha_n \in R$.

Given $r \in R$, we write $[r]_R$ to denote the situation in which r is secret-shared using our secret-sharing scheme from Construction 1 over the ring R (if R is obvious we omit it, as we have done until now). Recall from Sect. 2.2 that this means that there is a polynomial f over R of degree at most t such that party P_i has the share $r_i = f(\alpha_i)$ for $i = 1, \ldots, n$, and $r = f(0)$. We shall call this a *1D-sharing* of r, and refer to the shares $r_1, \ldots, r_n$ as level-one shares.

If each level-one share r_i is itself 1D-shared as $[r_i]_R$ we say we have a *2D-sharing* of r, and we denote this by $[\![r]\!]$. We refer to the entries of the share vector $[r_i]$ as *level-two shares.*

Finally, we denote by $\langle r \rangle$ the situation in which r is 2D-shared and additionally the parties hold authentication tags on r. We will call this a *2D*-sharing* of r, and it will be explained in detail in Sect. 4.5.

4.5 Sub-protocols for Secure Computation over Galois Rings

The overall protocol for secure computation follows the offline/online phase paradigm, which is typical from other secret-sharing based protocols, like these from [3–5,8,12,14]. Essentially, the parties preprocess some material in the offline phase which is used in the online phase to perform the computation, after sharing the inputs. The building blocks to achieve this include procedures for sharing values, generating signatures, checking correctness of triples, and some others. In this section we describe the pieces required to build our protocol, and also the protocol itself. We prove their security and analyze their communication complexity.

For the rest of the section we let κ denote the statistical security parameter.

Dispute Control Broadcast. This protocol allows a set of senders to broadcast a set of values among all the parties such that, with overwhelming probability, all the parties receive the same value which is the one sent initially if the sender is honest. Also, this broadcast is "compatible" with the dispute control mechanism, in the sense that it detects cheaters and generates new disputes. We remark that our model assumes a network with broadcast which may not provide dispute control by default.[2]

Even though the protocol for dispute control broadcast of [3] uses fields, no arithmetic properties of the input values are used. We may therefore just serialize elements of R as bit strings, map them to a finite field of suitable size, and use their protocol verbatim.

Complexity Analysis. The protocol communicates $O(\ell nd + \kappa n^2) = O(\ell n \log n + \kappa n^2)$ bits and broadcasts $O(n\kappa)$ bits. Here ℓ is the number of values in R being broadcasted, n is the number of players, and κ the security parameter.[3]

Verifiable 1D-Sharings. This protocol allows one party P_D to 1D-share some value $x \in R$ with the guarantee that the shares of the honest parties are consistent with a degree-t polynomial over R.[4] Note that we make no guarantees

[2] Assuming broadcast is necessary for $t \geq n/3$ since it is known that unconditional broadcast is not possible in this settting.

[3] Throughout this work we consider p and k as constants for the asymptotic complexity analysis. We also ignore the dispute control layer, as our complexity closely matches the one from [3] for the fault localization.

[4] Notice that if there are exactly $t + 1$ honest parties then this is trivial since any set of $t + 1$ values is consistent with a degree-t polynomial. However, VSS1D is needed for the general case.

beyond this; in particular, we do not guarantee robustness of shares. With the protocol, we can verify many different sharings at once, by opening a masked linear combination of the shares and checking correctness on the combination.

For this protocol we will make use of the packing technique as detailed in Sect. 4.1. Recall we move to an extension ring $\hat{R} \supset R$ with $\hat{R} = GR(p^k, d \cdot \hat{d})$. We denote a 1D-sharing over $\hat{R}$ as $[x]_{\hat{R}}$, which corresponds to sharing a vector of $\hat{d}$ elements of R via Lemma 4.

The protocol can be found in Fig. 3.

VSS1D

A party P_D will distribute ℓ values $a^{(1)}, \ldots, a^{(\ell)} \in R$ among all parties.

- P_D partitions $a^{(1)}, \ldots, a^{(\ell)} \in R$ into $L = \ell/\hat{d}$ vectors of length $\hat{d}$: $\mathbf{s}^{(j)} = (a^{(1,j)}, \ldots, a^{(\hat{d},j)}) \in R^{\hat{d}}$, for $j = 1, \ldots, L$.
- Let $s^{(j)} = \psi(\mathbf{s}^{(j)}) \in \hat{R}$ for $j = 1, \ldots, L$.

Private Computation: P_D samples at random $s^{(L+1)}, \ldots, s^{(L+n)} \in \hat{R}$ and deals $[s^{(1)}]_{\hat{R}}, \ldots, [s^{(L+n)}]_{\hat{R}}$ to all parties.

Fault Detection: Every verifier $P_V \in \mathcal{P} \setminus \mathcal{X}$ executes the following steps (in parallel).

1. P_V samples a challenge vector $(r_1, \ldots, r_L) \in \hat{R}^L$ and broadcasts this value using protocol DCBroadcast.
2. All the parties reconstruct $\sum_{i=1}^{L} r_i[s^{(i)}]_{\hat{R}} + [s^{(L+V)}]_{\hat{R}}$ towards P_V, who then checks correctness of the shares, i.e., P_V checks that these shares lie on a polynomial of degree at most t.
3. P_V broadcasts a bit indicating whether or not the check succeeded.

Fault Localization: See Section 3.2 of [3].

If no verifier P_V complained in the previous step, the output is defined to be $[a^{(1)}]_R, \ldots, [a^{(\ell)}]_R = \psi^{-1}([s^{(1)}]_{\hat{R}}), \ldots, \psi^{-1}([s^{(L)}]_{\hat{R}})$.

Fig. 3. Protocol for Verifiable Secret-Sharing

Proposition 4. *If the protocol VSS1D from Fig. 3 succeeds then, with probability at least $1 - p^{-\kappa}$, each $[a^{(m)}]_R$ is correctly 1D-shared for $m = 1, \ldots, \ell$. If the protocol aborts then a new dispute is generated. Input-privacy is guaranteed during the whole protocol (even if it fails).*

Proof. It is clear that the shared values remain secret since the random masks $s^{(L+V)}$ prevent them from being revealed.

Now, for soundness we consider the setting of an honest verifier P_V checking the shares of the dealer. Let C denote the $\hat{R}$-module of correct share vectors (see Construction 1). The adversary successfully cheats if for some i we have $[s^{(i)}] \notin C$ and the check passes, i.e. $\sum_{i=1}^{L} r_i[s^{(i)}]_{\hat{R}} + [s^{(L+V)}]_{\hat{R}} \in C$. Since the adversary knows which values they cheat on, we may take $i = 1$ without loss of generality. We can apply Lemma 3 and see that the probability of successfully cheating is at most $1/p^{d\hat{d}} \leq 1/p^{\kappa}$.

Finally, since each $[s^{(1)}]_{\hat{R}}, \ldots, [s^{(L)}]_{\hat{R}}$ is correctly shared, it follows from Lemma 4 that the shares $[a^{(1)},]_R \ldots, [a^{(\ell)}]_R$ output by the protocol are correct. For the case in which a dispute is generated see Lemma 2 in [3]. □

Complexity Analysis. The protocol communicates $O\left(n^2\kappa + \ell n \frac{\log n}{\kappa}\right)$ bits and broadcasts $O(n)$ bits.

Reconstruct 1D. Here we consider the setting in which a set of dealers $\mathcal{P}_D \subseteq \mathcal{P} \setminus \mathcal{X}$ have 1D-shared some values $[s^{(1,D)}], \ldots, [s^{(\ell,D)}]$, $P_D \in \mathcal{P}_D$. The goal is to reconstruct the values $s^{(m)} = \sum_{P_D \in \mathcal{P}_D} s^{(m,D)}$ for $m = 1, \ldots, \ell$ to a set of recipients $\mathcal{P}_R \subseteq \mathcal{P} \setminus \mathcal{X}$. This is achieved by letting each player $P_i \in \mathcal{P}$ compute its share of the sum $s_i^{(m)} = \sum_{P_D \in \mathcal{P}_D} s_i^{(m,D)}$ and send it to each player in $\mathcal{P}_R$. Then a dispute control layer makes sure that all parties agree that the reconstruction was done successfully.

Proposition 5. *There is a protocol* Reconstruct1D *such that, on input some values* $[s^{(1,D)}], \ldots, [s^{(\ell,D)}]$ *correctly shared by each* $P_D \in \mathcal{P} \setminus \mathcal{X}$*, the protocol either fails or each party in* $\mathcal{P} \setminus \mathcal{X}$ *receives* $s^{(m)} = \sum_{P_D \in \mathcal{P}_D} s^{(m,D)}$ *for* $m = 1, \ldots, \ell$*. Moreover, if the protocol aborts a new pair of players in dispute is identified.*

For the description of the protocol and its proof of security see Lemma 3 in Sect. 3.2 of [3]. The main observation is that their argument applies directly to our setting since it only relies on polynomial interpolation, which works for R in essentially the same way as it does for a field as long as the base points are chosen to form an exceptional sequence.

Complexity Analysis. The protocol communicates $O(\ell n^2 d)$ bits and broadcasts $O(nd)$ bits, where d is the degree of the Galois ring R over $\mathbb{Z}/p^k\mathbb{Z}$.

Generating Random Challenges. An essential tool needed for statistically secure MPC is the generation of publicly known random elements. This is achieved by a protocol GenerateChallenges which operates as follows.

1. Each party $P_i \in \mathcal{P} \setminus \mathcal{X}$ samples some random values $s^{(1,i)}, \ldots, s^{(\ell,i)} \in R$ and uses VSS1D to distribute correct shares of it.
2. The parties compute $[s^{(m)}] = \sum_{P_i \in \mathcal{P} \setminus \mathcal{X}} [s^{(m,i)}]$ and open $s^{(m)}$ to all parties in $\mathcal{P} \setminus \mathcal{X}$ using Reconstruct1D, for $m = 1, \ldots, \ell$.

Since the additive group of R is abelian, if each $s^{(m,i)}$ is independent and there is at least one that is uniformly random, then $s^{(m)}$ is random. Now, the $s^{(m,i)}$ are independent from each other since they are secret-shared, so one player cannot choose its share conditioned on the other players' shares.

Complexity Analysis. The protocol communicates $O(n^3\kappa + \ell n^2 d)$ bits and broadcasts $O(nd)$ bits where d is the degree of the Galois ring R over $\mathbb{Z}/p^k\mathbb{Z}$.

Upgrading 1D-Sharings to 2D-Sharings. The goal of this protocol is to upgrade a 1D-sharing $[a]$ of $a \in R$ to a 2D-sharing $[\![a]\!]$. In fact, several values $a^{(1)}, \ldots, a^{(\ell)} \in R$ will be upgraded in one go, and moreover, sums of 1D-shares instead of individual 1D-shares will be upgraded due to our use-case.

More precisely, let $\mathcal{P}_D \subseteq \mathcal{P} \setminus \mathcal{X}$ be some subset of dealers. Each $P_D \in \mathcal{P}_D$ has a list of values $a^{(1,D)}, \ldots, a^{(\ell,D)} \in R$ it has secret-shared. The goal of the Upgrade1Dto2D sub-protocol is to let each party P_i distribute shares of its share $a_i^{(m)}$ of $a^{(m)} = \sum_{P_D \in \mathcal{P}_D} a^{(m,D)}$ for $m = 1, \ldots, \ell$. At the end of the protocol it is guaranteed that all shares (both, the shares of each $a^{(m)}$ and the shares of their shares) are correct.

Upgrade1Dto2D

Let $a^{(1,D)}, \ldots, a^{(\ell,D)} \in R$ such that each $a^{(m,D)}$ has been 1D-shared by $P_D \in \mathcal{P}_D$.

- The parties partition $[a^{(1,D)}]_R, \ldots, [a^{(\ell,D)}]_R$ into $L = \ell/\hat{d}$ vectors of length $\hat{d}$: $\mathbf{s}^{(j,D)} = ([a^{(1,j,D)}]_R, \ldots, [a^{(\hat{d},j,D)}]_R) \in R^{\hat{d}}$, for $j = 1, \ldots, L$.
- Let $[s^{(j)}]_{\hat{R}} = \sum_{P_D \in \mathcal{P}_D} \psi(\mathbf{s}^{(j,D)}) \in \hat{R}$ for $j = 1, \ldots, L$.

Private Computation: 1. Each $P_D \in \mathcal{P}_D$ shares a random value $s^{(L+1,D)} \in \hat{R}$.
2. Each player P_i 1D-shares each of its shares $s_i^{(m)} \in \hat{R}$ for $m = 1, \ldots, L+1$. We denote by $s_{ij}^{(m)} \in \hat{R}$ the share of $s_i^{(m)}$ received by P_j.

Fault Detection: Using the protocol GenerateChallenges, the parties jointly generate random values $(r_1, \ldots, r_L) \in \hat{R}^L$. Then the following is executed for every verifier $P_V \in \mathcal{P} \setminus \mathcal{X}$.
1. Every P_j with $P_j \leftrightarrow P_V$ computes the share $s_{ij} = \sum_{m=1}^{L} r_m \cdot s_{ij}^{(m)} + s_{ij}^{(L+1)}$ for every P_i with $P_i \leftrightarrow P_j$, and sends these to P_V (notice that these are shares of $s_i = \sum_{m=1}^{L} r_m \cdot s_i^{(m)} + s_i^{(L+1)}$).
2. For every P_i with $P_i \leftrightarrow P_V$, P_V checks that $(s_{i1}, \ldots, s_{in})$ lie in a polynomial over $\hat{R}$ of degree at most t. Then broadcasts accept or reject depending on the case.
3. If P_V accepted in the previous step, then interpolate $s_1, \ldots, s_n$ and check whether or not these lie in a polynomial of degree at most t.

Fault Localization: See protocol Upgrade1Dto2D in Section 3.4 of [3].

If no verifier P_V complained in the previous step, the output is defined to be $[\![a^{(1)}]\!]_R, \ldots, [\![a^{(\ell)}]\!]_R = \psi^{-1}([\![s^{(1)}]\!]_{\hat{R}}), \ldots, \psi^{-1}([\![s^{(L)}]\!]_{\hat{R}})$.

Fig. 4. Protocol for upgrading 1D-shares to 2D-shares

Proposition 6. *If Upgrade1Dto2D aborts, then a new conflicting pair of parties is detected. Otherwise, it is guaranteed with probability at least* $1 - p^{-d}$ *that the values* $s^{(m)} \in R$ *for* $m = 1, \ldots, \ell$ *are correctly 2D-shared, meaning that for each* m *there are polynomials* $f^{(m)}, f_1^{(m)}, \ldots, f_n^{(m)} \in R[X]$ *of degree at most* t *such that each party* P_j *has shares* $s_j^{(m)}, s_{ij}^{(m)} \in R$ *with* $s_j^{(m)} = f^{(m)}(j)$, $s_{ij}^{(m)} = f_i^{(m)}(j)$, $s_i^{(m)} \equiv_k f_i^{(m)}(0)$ *and* $s^{(m)} = f^{(m)}(0)$.

Proof. The proof of this proposition follows the lines of the proof of Proposition 4. □

Complexity Analysis. The protocol communicates $O(n^3\kappa + \ell n^2)$ bits and broadcasts $O(n\kappa)$ bits.

Information-Checking Signatures with Dispute Control. The goal of information-checking signatures, or IC signatures for short, is to provide a way for one party P_R to prove to another party P_V that it received some specific shares from some other party P_S. This will be used in the online phase to detect cheaters when revealed shares happen to be inconsistent. The idea is that whenever a player sends his share, he is "committed" to it by means of the authentication tags and therefore, if he sends an incorrect share, this can be detected by checking the tags.

For the IC signatures in this work we follow a similar approach to [3], which at a very high level consists of finding a polynomial f that interpolates a set of messages as well as the point $(0, y)$ for a randomly chosen y. The value y will be referred to as the *authentication tag*. The *authentication key* will be a random poiont $(u, f(u))$ on this polynomial where u is not an evaluation point corresponding to any of the messages. To check correctness, the key is used to interpolate the polynomial and then it is checked that its evaluation at zero matches the presented tag. Intuitively, if any message is modified then the polynomial will be different, and the only way in which an attacker can make the check pass is by presenting the right tag, which is equivalent to guessing point used as authentication key. If there are enough points to choose from, this happens only with low probability.

In more detail, the protocol IC-Distr allows a sender P_S to send ℓ values $m_1, \ldots, m_\ell \in R$ to a receiver P_R along with authentication tags, and to send an authentication key to a verifier P_V. At a later point the protocol IC-Reveal can be called to verify correctness of these tags. In this protocol, party P_R sends the messages and their tags to P_V, who can then verify their correctness using its authentication key.

Theorem 7 (Lemma 6 from [3]). *If IC-Distr succeeds and P_V, P_R are honest, then with overwhelming probability P_V accepts the message m in IC-Reveal (completeness). If IC-Distr fails, then the localized pair in dispute contains at least one corrupted player. If P_S and P_V are honest, then with overwhelming probability, P_V rejects any fake message $m' \neq m$ in IC-Reveal (correctness). If P_S and P_R are honest, then P_V obtains no information about m in IC-Distr (even if it fails) (privacy).*

IC-Distr

A sender P_S has ℓ messages $m^{(1)}, \dots, m^{(\ell)} \in R$.

- Let $\hat{d}$ be such that $d \cdot \hat{d} \geq \kappa$. let $L = \ell/\hat{d}$, and assume that $\hat{d}$ is large enough so $p^\kappa \geq L + \kappa + 1$, i.e. $\hat{d} \geq \ell/(p^\kappa - \kappa - 1)$.
- P_S partitions $m^{(1)}, \dots, m^{(\ell)} \in R$ into $\ell/\hat{d}$ vectors of length $\hat{d}$: $\mathbf{s}^{(j)} = (m^{(1,j)}, \dots, m^{(\hat{d},j)}) \in R^{\hat{d}}$, for $j = 1, \dots, L$.
- Let $s^{(j)} = \psi(\mathbf{s}^{(j)}) \in \hat{R}$ for $j = 1, \dots, L$.

Private Computation: 1. Let $B = \{\beta_1, \dots, \beta_L\} \subseteq \hat{R}$ be an exceptional sequence. P_S selects κ random authentication tags $y_1, \dots, y_\kappa \in \hat{R}$ and random points $u_1, \dots, u_\kappa \in \hat{R} \setminus (B \cup \{0\})$ such that $B \cup \{0, u_1, \dots, u_\kappa\} \subseteq \hat{R}$ forms an exceptional sequence.
2. For $i = 1, \dots, \kappa$, P_S computes the polynomial f_i over $\hat{R}$ of degree at most L interpolating $(0, y_i), (\beta_1, s^{(1)}), \dots, (\beta_L, s^{(L)})$, and computes $v_i = f_i(u_i)$.
3. P_S sends the messages $m^{(1)}, \dots, m^{(\ell)}$ to P_R, along with the authentication tags $y_1, \dots, y_k$. It also sends the authentication keys $(u_1, v_1), \dots, (u_\kappa, v_\kappa)$ to P_V.

Fault Detection: P_V reveals a random half of the keys to P_R. Then P_R checks the validity of these keys, who then broadcast accept or reject depending on the case. If the check passes then the remaining, unrevealed half of the keys is kept as the actual keys.

Fault Localization: See Section 3.5 of [3].

Fig. 5. Protocol for Distributing IC Signatures

IC-Reveal

A receiver P_R has ℓ messages $m^{(1)}, \dots, m^{(\ell)} \in R$ and $\kappa' = \kappa/2$ authentication tags $y_1, \dots, y_{\kappa'} \in \hat{R}$. A verifier P_V has κ' authentication keys $(u_1, v_1), \dots, (u_{\kappa'}, v_{\kappa'})$ corresponding to these messages.

- P_R partitions $m^{(1)}, \dots, m^{(\ell)} \in R$ into $L = \ell/\hat{d}$ vectors of length $\hat{d}$: $\mathbf{s}^{(j)} = (m^{(1,j)}, \dots, m^{(\hat{d},j)}) \in R^{\hat{d}}$, for $j = 1, \dots, L$.
- Let $s^{(j)} = \psi(\mathbf{s}^{(j)}) \in \hat{R}$ for $j = 1, \dots, L$.

1. P_R sends the messages and the authentication tags to P_V
2. P_V checks the validity of the tags using its authentication keys by checking that, for at least one i, the points $(0, y_i), (\beta_1, s^{(1)}), \dots, (\beta_L, s^{(L)}), (u_i, v_i)$ lie on a polynomial of degree at most L over $\hat{R}$.

Fig. 6. Protocol for Revealing and Checking IC Signatures

Proof. Regarding completeness, notice that if the randomly chosen $\kappa/2$ tags are correct, then it holds that at least one of the remaining authentication tags is valid with probability at least $1 - \kappa/2^\kappa$.

For correctness, consider the scenario of an honest P_V and a corrupt P_R. Suppose that P_R manages to make the check pass whilst presenting a different set of messages. Let f_i be the polynomial of degree at most L over $\hat{R}$ interpolating $(\beta_1, s'^{(1)}), \ldots, (\beta_L, s'^{(L)}), (u_i, v_i)$, then P_R must have sent a tag y'_i that is equal to one of the elements in $\{f_1(0), \ldots, f_\kappa(0)\}$. This can be done only if P_R guesses at least one of the authentication keys (u_i, v_i). Recall that $\hat{R}$ has a Lenstra constant of at least p^κ, so there are at least $p^\kappa - L - 1$ possibilities for each u_i. This means that the probability of guessing *at least one* u_i is at most $\kappa/(p^\kappa - L - 1)$.

For the proof of the other properties see the proof of Lemma 6 in [3]. □

Upgrading 2D-Sharings to 2D*-Sharings. Recall that in Sect. 4.4 we mentioned the concept of 2D*-shares, but we did not explicitly define it since we did not have the concept of IC signatures. We begin by defining what a 2D*-share is. Given $a \in R$, we say that a is 2D*-shared, written as $\langle a \rangle$, if it holds that $[\![a]\!]$ and also, for every set of three players P_R, P_S, P_V such that $P_R \leftrightarrow P_S, P_S \leftrightarrow P_V$ and $P_R \leftrightarrow P_V$ it holds that P_R has authentication tags of the level-two share of P_S's share, and P_V has the corresponding authentication keys.

Protocol Upgrade2Dto2D* takes as input some 2D-shared values $s^{(1)}, \ldots, s^{(\ell)} \in R$, and upgrades them to 2D*-shares. The protocol works by calling IC-Distr for every set of three players P_R, P_S, P_V such that $P_R \leftrightarrow P_S, P_S \leftrightarrow P_V$ and $P_R \leftrightarrow P_V$, where the message m are the shares $s_{SR}^{(1)}, \ldots, s_{SR}^{(\ell)}$.

For the dispute control layer of the protocol and its security proof see Sect. 3.6 of [3].

Complexity Analysis. The protocol communicates $O(\kappa^2 n^3)$ bits and broadcasts $O(n\kappa)$ bits.

Triple-Checking Protocol. The protocol SacrificeTriple, described in Fig. 7, allows the parties to check that some given shares $[a], [b], [c]$ satisfy $c = a \cdot b$. This is achieved by generating some shares $[a'], [c']$ where $c' = a' \cdot b$, and "sacrificing" $([a'], [b], [c'])$ to check correctness of $([a], [b], [c])$.

For the security of the SacrificeTriple protocol we need to argue about the number of roots of a polynomial over a ring. In general, not much can be said since over a ring with zero divisors a polynomial can have many more roots than its degree. However, we have the following lemma, which bounds the number of roots that constitute an exceptional sequence.

SacrificeTriple

The inputs are 1D-shared values $[a_k^{(m)}]_R, [b_k^{(m)}]_R, [c^{(m,k)}]_R$ for $m = 1, \ldots, \ell$, where $a_k^{(m)}, b_k^{(m)}, c^{(m,k)}$ were dealt by party $P_k \in \mathcal{P} \setminus \mathcal{X}$.

1. Every player $P_k \in \mathcal{P} \setminus \mathcal{X}$ verifiably 1D-shares random values $\bar{a}_k^{(m)} \in \hat{R}$ and $\bar{c}^{(m,k)} \in \hat{R}$ with $\bar{c}^{(m,k)} = \bar{a}_k^{(m)} \cdot b_k^{(m)}$ for $m = 1, \ldots, \ell$ as follows:
   (a) For $m = 1, \ldots, \ell$, player $P_k \in \mathcal{P} \setminus \mathcal{X}$ samples $\bar{a}_k^{(m)}$ and $\bar{c}^{(m,k)}$ as specified above. Let $\psi^{-1}(a_k^{(m)}) = (\bar{a}_{k,1}^{(m)}, \ldots, \bar{a}_{k,\hat{d}})^{(m)} \in R^{\hat{d}}$ and $\psi^{-1}(\bar{c}^{(m,k)}) = (\bar{c}_1^{(m,k)}, \ldots, \bar{c}_{\hat{d}}^{(m,k)}) \in R^{\hat{d}}$.
   (b) P_k 1D-shares the $2\ell\hat{d}$ values $\bar{a}_{k,1}^{(m)}, \ldots, \bar{a}_{k,\hat{d}}^{(m)} \in R$ and $\bar{c}_1^{(m,k)}, \ldots, \bar{c}_{\hat{d}}^{(m,k)} \in R$ using the VSS1D protocol, for $m = 1, \ldots, \ell$. This implies that $\bar{a}_k^{(m)} \in \hat{R}$ and $\bar{c}^{(m,k)}\hat{R}$ are verifiably 1D-shared over $\hat{R}$.
2. Parties jointly sample a random value $r \in \hat{R}$ using protocol GenerateChallenges.
3. Each player $P_k \in \mathcal{P} \setminus \mathcal{X}$ sends $\tilde{a}_k^{(m)} = r \cdot a_k^{(m)} + \bar{a}_k^{(m)} \in \hat{R}$ to all parties P_i with $P_i \leftrightarrow P_k$, for $m = 1, \ldots, \ell$.
4. Parties jointly sample a random value $s \in \hat{R}$ using protocol GenerateChallenges.[a]
5. Parties invoke Reconstruct1D to reconstruct $[z^{(k)}]_{\hat{R}} = \sum_{m=1}^{\ell} s^{m-1}[z_k^{(m)}]_{\hat{R}}$, where $[z_k^{(m)}]_{\hat{R}} = \tilde{a}_k^{(m)}[b_k^{(m)}]_R - r[c^{(m,k)}]_R - [\bar{c}^{(m,k)}]_R$, for $k = 1, \ldots, n$.[b]
6. The parties check that $z^{(k)} = 0$ for all k. If this fails for some k_0 then new disputes $P_i \not\leftrightarrow P_{k_0}$ are generated for all $P_i \in \mathcal{P} \setminus \mathcal{X}$.

[a] We could choose ℓ independent challenges instead, but we use this optimization to save in communication. Notice that a similar optimization can be applied to the protocol from [3]

[b] Some extra step is needed to ensure players are committed to their $\tilde{a}$. See [3] for the details.

Fig. 7. Protocol for Verifying Multiplications

Lemma 5. *Let $f(X) \in R[X]$ be a non-zero polynomial of degree at most ℓ. If $\{\alpha_1, \ldots, \alpha_m\} \subseteq R$ are different roots of f that form an exceptional sequence, then $m \leq \ell$.*

Proof. This follows from Theorem 3. Suppose that $\ell < m$, so $\ell \leq m - 1$. We know that there is a unique polynomial of degree at most $m-1$ that interpolates the points $(\alpha_1, 0), \ldots, (\alpha_m, 0)$, but both the zero polynomial and f satisfy this condition, so f is the zero polynomial, which is a contradiction. Therefore, we conclude that $m \leq \ell$. □

We proceed to the proof of security of the protocol SacrificeTriple.

Proposition 7. *Assume all shares* $[a_k^m], [b_k^m], [c^{(m,k)}]$ *are correctly 1D-shared. If the protocol* SacrificeTriple *succeeds, then with probability at least* $1-\ell/p^\kappa$ *it holds that* $c^{(m.k)} = a_k^m \cdot b_k^m$ *for all* $P_k \in \mathcal{P} \setminus \mathcal{X}$ *and* $m = 1, \ldots, \ell$*. If the protocol aborts then it generates a new dispute.*

Proof. Consider a corrupt player P_k for which $\bar{c}^{(m,k)} = \bar{a}_k^{(m)} b_k^{(m)} + \gamma_k^{(m)}$ and $c^{(m,k)} = a_k^{(m)} b_k^{(m)} + \delta_k^{(m)}$ with $\delta_k^{(m)} \neq 0$ for some m, say $m = 1$. Now, suppose the protocol succeeds, then $z^{(k)} = 0$. However, we see that this value is equal to[5]

$$\begin{aligned} z^{(k)} &= \sum_{m=1}^{\ell} s^{m-1}(\tilde{a}_k^{(m)} b_k^{(m)} - r c^{(m,k)} - \bar{c}^{(m,k)}) \\ &= \sum_{m=1}^{\ell} s^{m-1}((r a_k^{(m)} + \bar{a}_k^{(m)}) b_k^{(m)} - r(a_k^{(m)} b_k^{(m)} + \delta_k^{(m)}) - (\bar{a}_k^{(m)} b_k^{(m)} + \gamma_k^{(m)})) \\ &= -\sum_{m=1}^{\ell} s^{m-1}(r\delta_k^{(m)} + \gamma_k^{(m)}) = 0. \end{aligned}$$

Now, since $\delta_k^{(1)} \neq 0$, it follows from Lemma 2 that $r\delta_k^{(1)} + \gamma_k^{(1)}$ is non-zero with probability at least $1 - p^{-\kappa}$ (over the choice of r).

Applying Lemma 5 we see that conditioned on $r\delta_k^{(1)} + \gamma_k^{(1)} \neq 0$ the event $r\delta_k^{(1)} + \gamma_k^{(1)} + \sum_{m=2}^{\ell} s^{m-1}(r\delta_k^{(m)} + \gamma_k^{(m)}) = 0$ holds with probability at most $(\ell-1)/p^\kappa$. Furthermore, using the same lemma we see that the probability that $r\delta_k^{(1)} + \gamma_k^{(1)} = 0$ is at most $1/p^\kappa$. Therefore, putting these together we obtain that the adversary can only successfully cheat with probability at most ℓ/p^κ.

Regarding privacy, we observe that the value $r \cdot a_k^{(m)} \in \hat{R}$, which contains information about $a_k^{(m)}$, is masked by the element $\bar{a}_k^{(m)} \in \hat{R}$. Since this element is uniformly random for an honest P_k, and given that $\hat{R}$ is an additive group, we conclude that the private value $a_k^{(m)}$ of P_k remains hidden.

For the arguments related to dispute control see Lemma 8 in [3]. $\square$

Complexity Analysis. Assuming that $n\log(n) \leq \kappa^2$, the protocol transmits $O(n^3\kappa + n^2\kappa\ell)$ bits, and broadcasts $O(n\kappa)$ bits.

4.6 Final Protocol

Offline Phase. In the offline phase the parties generate a number M of multiplication triples $(\langle a\rangle, \langle b\rangle, \langle c\rangle)$, where $c = a \cdot b$ and a, b are random. This phase is totally independent of the circuit to be computed (parties only need to make sure to generate as many triples as multiplication gates in the circuit), and therefore it can be executed at a totally different time than the evaluation of the circuit itself, thus the name "offline".

To compute these sharings, a technique known as re-sharing is used to obtain $[a \cdot b]$ from $[a]$ and $[b]$. This works by letting the parties locally compute degree $2t-$sharings of $a \cdot b$ by taking the local product of their shares on a and b.

[5] We use the equality $\tilde{a}_k^{(m)} = r a_k^{(m)} + \bar{a}_k^{(m)}$, which follows from the extra step we omitted in the protocol.

Then these shares are distributed and an appropriate linear combination is taken to obtain $[a \cdot b]$.

Assume for simplicity that n^2 divides M. To produce the M triples, the parties produce n^2 batches of $L = M/n^2$ triples each. To generate the L triples of each batch (or *segment*), the parties run the protocol from Fig. 8. Notice that each segment may fail due to the dispute control, in which case a new dispute is identified and the segment must be repeated. Since there are most n^2 different disputes that can occur, there may be up to n^2 repetitions of segments overall, and since there are at most n^2 segments we see that there are at most $2n^2$ segment executions.

Proposition 8. *The preprocessing protocol generates correctly 2D*-shared multiplication triples with overwhelming probability.*

Proof. The proof follows from the properties of Upgrade1Dto2D, VSS1D and Upgrade2Dto2D*. See Lemma 10 in [3] for the details. □

Complexity Analysis. Suppose that there are M triples to be processed. The preprocessing phase communicates $O(Mn^2 \log n + \kappa^2 n^5)$ bits and broadcasts $O(n^3\kappa)$ bits.

Preprocessing Protocol

Since this is the first protocol to be executed, initially the dispute set and the set of identified corrupt parties are $\Delta, \mathcal{X} = \{\}$. The following is executed for each segment, and each time a new dispute pair $P_i \not\leftrightarrow P_j$ is identified, it is added to Δ and the segment is repeated.

1. Each player P_k 1D-shares $2L$ random values $a^{(m,k)}, b^{(m,k)} \in R$ for $m = 1, \ldots, L$.
2. Upgrade1Dto2D is called on $a^{(m,i)}$ for $m = 1, \ldots, L$ and $P_i \in \mathcal{P} \setminus \mathcal{X}$ to obtain correct 2D-shares $[\![a^{(m)}]\!]$ and $[\![b^m]\!]$ for $m = 1, \ldots, L$, where $a^{(m)} = \sum_{P_i \in \mathcal{P} \setminus \mathcal{X}} a^{(m,i)}$ and similarly $b^{(m)} = \sum_{P_i \in \mathcal{P} \setminus \mathcal{X}} b^{(m,i)}$.
3. The players invoke VSS1D to let each $P_k \in \mathcal{P} \setminus \mathcal{X}$ 1D-share the values $c^{(m,k)} = a_k^{(m)} \cdot b_k^{(m)}$ for $m = 1, \ldots, L$.
4. Invoke the protocol SacrificeTriple to prove that the value $[c^{(m,k)}]$ shared on the previous step is the product of $[a_k^{(m)}]$ and $[b_k^{(m)}]$ (recall that $a^{(m)}$ and $b^{(m)}$ are 2D-shared), for $m = 1, \ldots, L$.
5. Let $\lambda_1, \ldots, \lambda_n \in R$ be such that $f(0) = \sum_{i=1}^{n} \lambda_i \cdot f(i)$ for any polynomial f over R of degree at most $2t$. The parties use Upgrade1Dto2D to compute $[\![c^{(m)}]\!] \leftarrow \sum_{k=1}^{n} \lambda_k \cdot [c^{(m,k)}]$ for $m = 1, \ldots, L$.
6. Parties use Upgrade2Dto2D* to upgrade all shares to 2D*-shares.

Fig. 8. Protocol for Preparing Multiplication Triples

Online Phase. In the online phase is where the parties actually compute the circuit securely, using the triples that were preprocessed in the offline phase. We present here the online phase without the dispute control layer, which takes care of executing only certain amount of steps within a segment and checking correctness within that segment, repeating it if something was found to be inconsistent. We refer the reader to [3] for the details of how this is done.

This phase starts by the parties sharing their inputs. This is done by letting P_i, for each i, share its input $s^{(i)} \in R$ to the other parties. For this P_i begins by 1D-sharing $s^{(i)}$ and then the parties invoke the procedures Upgrade1Dto2D and Upgrade2Dto2D* to obtain 2D*-sharings of $s^{(i)}$. Then the parties process the gates in topological order. For the addition gates, all the 2D-shares of the inputs are simply added locally, thus requiring no interaction. However, when two shared values $[\![x]\!]$ and $[\![y]\!]$ need to be multiplied, the parties must make use of a preprocessed triple $([\![a]\!], [\![b]\!], [\![c]\!])$ with $c = a \cdot b$. The multiplication is then achieved by computing $[\![x - a]\!] = [\![x]\!] - [\![a]\!]$ and opening it as ϵ, and similarly $[\![y - b]\!] = [\![y]\!] - [\![b]\!]$ and opening it as δ, and then computing $[\![x \cdot y]\!] = [\![c]\!] + \delta[\![x]\!] + \epsilon[\![y]\!] + \epsilon\delta$.

As we mentioned at the beginning of the section, the details about how to handle consistency are exactly the same as discussed in [3], so we omit some of the details of such procedure. See Sect. 6 in the aforementioned reference to see how this is done precisely. Something to point out is that consistency is eventually checked by means of the IC signatures from Sect. 4.5. This tool is used in dispute control so that some party P_S can prove to some verifier P_V that certain values were indeed sent by some other party P_R.

Complexity Analysis. The input phase communicates $O(c_I n^2 \log n + \kappa n^5)$ bits where c_I is the number of input gates, and broadcasts $O(\kappa n^3)$ bits. The computation phase communicates $O(|C| n^2 \log n + n^4 \kappa^2)$ bits where $|C|$ is the size of the circuit, and broadcasts $O(n^3 \kappa)$ bits.

4.7 Computation over $\mathbb{Z}/p^k\mathbb{Z}$

Summing up, we have seen so far how to perform unconditional secure computation over the Galois ring $R = (\mathbb{Z}/p^k\mathbb{Z}[Y])/(h(Y))$. However, we wish to obtain unconditional secure computation over $\mathbb{Z}/p^k\mathbb{Z}$ itself. We can embed $\mathbb{Z}/p^k\mathbb{Z}$ into R in the natural way, and as seen in Sect. 3.4 this works for passive adversaries, but if an active adversary manages to share values that are in $R \setminus \mathbb{Z}/p^k\mathbb{Z}$, correctness and security could be broken. As discussed in Sect. 3.4 and Sect. 4.1 this reduces to securely sampling an R-sharing of a random element $[s]$ where $s \leftarrow \mathbb{Z}/p^k\mathbb{Z}$.

Here we present a protocol RandElStat(S) in Fig. 9 for sampling this element $[s] \in S$ efficiently. Here $S \subseteq R$ denotes an arbitrary subring; for our use case $S = \mathbb{Z}/p^k\mathbb{Z}$. We have made the protocol to be explicit and removed any mention of tensor products, but the intuition for this was given already in Sect. 4.1. The protocol succeeds with overwhelming probability.

RandElStat(S)

OUTPUT: sharings $[x_j^{(i)}]$ for $j = 0, \ldots, d-1$ and $i = 1, \ldots, L$ for a total of dL random elements, where the shares are in R and the secrets $x_j^{(i)}$ are in S.
PUBLIC INFORMATION: fix $\xi \in R$ such that $\{1, \xi, \xi^2, \ldots, \xi^{d-1}\}$ is an S-basis for R as an S-module. With respect to this basis, multiplication by an element $r \in R$ can be represented by a $d \times d$ matrix M_r with entries in S.

Private Computation: Each player $P_k \in \mathcal{P} \setminus \mathcal{X}$ samples $d(L+1)$ uniformly random values $x_j^{(i,k)} \leftarrow S$ for $j = 0, \ldots, d-1$ and $i = 1, \ldots, L+1$, and 1D-shares each of them over R. The players compute $[x_j^{(i)}] = \sum_{P_k \in \mathcal{P} \setminus \mathcal{X}}^{n} [x_j^{(i,k)}]$

Fault Detection: The players run GenerateChallenges to sample uniformly random $r_1, \ldots, r_L$ in $\hat{R}$, with associated matrices as mentioned above. Then the following is executed for every verifier $P_V \in \mathcal{P} \setminus \mathcal{X}$.

1. The players interpret the random elements $[x_j^{(i)}]$ as $L+1$ column vectors of length d, i.e. for each $i = 1, \ldots, L+1$ we have $[\mathbf{x}^{(i)}] = ([x_0^{(i)}], \ldots, [x_{d-1}^{(i)}])^T$. Then, they compute the sum $[\mathbf{y}] = M_{r_1}[\mathbf{x}^{(1)}] + \cdots + M_{r_L}[\mathbf{x}^{(L)}] + [\mathbf{x}^{(L+1)}]$ and send the shares of $\mathbf{y}$ to P_V.
2. P_V checks if it holds that all the entries of $\mathbf{y}$ are in S, and broadcast a bit indicating which is the case.

If all verifiers $P_V \in \mathcal{P} \setminus \mathcal{X}$ accepted in the previous step then output the shares $[x_j^{(i)}]$.

Fault Localization: Run the following for the smallest $P_V \in \mathcal{P} \setminus \mathcal{X}$ that complained in the fault detection phase.

1. Every player P_k with $P_k \leftrightarrow P_V$ sends their shares of each $x_j^{(i,\ell)}$ to P_V, for $j = 0, \ldots, d-1$, $i = 1, \ldots, L$ and $P_\ell \leftrightarrow P_k$.
2. P_V checks that all the shares for $P_\ell \leftrightarrow P_V$ interpolate correctly.
3. If they do interpolate correctly then P_V gets $x_j^{(i,\ell)}$ for $j = 0, \ldots, d-1$, $i = 1, \ldots, L$ and $P_\ell \in \mathcal{P} \setminus \mathcal{X}$. P_V broadcasts the smallest index ℓ of the party for which $x_j^{(i,\ell)} \notin S$ and the protocol fails with $P_V \not\leftrightarrow P_k$.[a]
4. If they do not interpolate correctly then P_V broadcasts the smallest indexes ℓ, i, j for which interpolation of $x_j^{(i,\ell)}$ failed.
5. Each party $P_k \in \mathcal{P} \setminus \mathcal{X}$ with $P_k \leftrightarrow P_\ell$ broadcasts its share of $x_j^{(i,\ell)}$.
6. If the broadcasted shares interpolate correctly then P_V broadcasts the index k of a party P_k with $P_k \leftrightarrow P_V$ that broadcasted a share different than the one it sent to P_V before and the protocol fails with $P_V \not\leftrightarrow P_k$.
7. Otherwise, the accused party P_ℓ broadcasts the index of the party P_k who broadcasted a wrong share and the protocol fails with $P_\ell \not\leftrightarrow P_k$.

[a] Such party exists with overwhelming probability, as we argue in Proposition 9

Fig. 9. Statistically secure protocol for generating sharings of random elements in a Galois subring $S \subset R$

With this protocol in hand, the input phase from the previous section is modified slightly in order to make sure that underlying inputs lie in $\mathbb{Z}/p^k\mathbb{Z}$. This is done as follows:

1. Party $P_i \in \mathcal{P} \setminus \mathcal{X}$ shares its input $x \in \mathbb{Z}/p^k\mathbb{Z}$ as $[x]_R$.
2. The parties use $\mathsf{RandElStat}(\mathbb{Z}/p^k\mathbb{Z})$ to obtain shares $[s]_R$ of a random element $s \in \mathbb{Z}/p^k\mathbb{Z}$. Then use $\mathsf{Reconstruct1D}$ to open $[s+x]_R$.
3. If $s + x \notin \mathbb{Z}/p^k\mathbb{Z}$ then add $P_i \in \mathcal{X}$, i.e. mark P_i as corrupt.

It is clear that if the check is sound since $x \notin \mathbb{Z}/p^k\mathbb{Z}$ iff $s + x \notin \mathbb{Z}/p^k\mathbb{Z}$. Regarding the security of $\mathsf{RandElStat}$, we have the following proposition.

Proposition 9. *If RandElStat succeeds, then, with probability at least* $1 - p^{-\kappa}$*, each value* $s_j^{(i)}$ *is uniformly random in* S*. If it fails then a new dispute pair is generated.*

Proof. Suppose the check succeeds for an honest verifier P_V and the adversary cheats successfully, i.e. there is an element $x_j^{(i_*)}$ which is not in S. Recall $\{1, \xi, \ldots, \xi^{d-1}\}$ is an S-basis for R, so we may without loss of generality assume that the ξ^m-coefficient of $x_j(i_*)$ is non-zero. We have

$$[\mathbf{y}] = M_{r_1}[\mathbf{x}^{(1)}] + \cdots + M_{r_L}[\mathbf{x}^{(L)}] + [\mathbf{x}^{(L+1)}] \tag{1}$$

where each element of $\mathbf{y}$ is in S, but note that the shares are actually vectors in R^d. On both sides of Eq. 1, we first take the coefficients of ξ^m *for each* R-element, and then interpret the resulting S-vectors and matrices M_r as elements of R. Both of these operations are S-linear. The result is the equation $0 = r_1u_1 + \cdots + r_Lu_L + u_{L+1}$, where $u_i = \phi\left(x_0^{(i)}\right) + \phi\left(x_1^{(i)}\right)\xi + \cdots + \phi\left(x_{d-1}^{(i)}\right)\xi^{d-1}$ for each i, and $\phi : R \to S$ maps an element in R to its coefficient of ξ^m. Similarly to the proof of Proposition 4, we apply Lemma 2 to conclude that this equation holds with probability at most p^{-d}, since each r_i is uniformly random. □

5 Conclusions

In this work, we have answered the open question "*Can we design protocols that work directly over* $\mathbb{Z}/p^k\mathbb{Z}$*?*" in the affirmative. We have developed novel machinery that allows us to adapt existing protocols for information-theoretic MPC to work over the ring $\mathbb{Z}/p^k\mathbb{Z}$, for any prime p and any positive integer k. In fact, by using CRT, this implies information-theoretic MPC over the ring $\mathbb{Z}/N\mathbb{Z}$ for *any* integer N. The communication complexity of our techniques introduce an overhead of only $\log n$ compared to the corresponding protocols over fields, where n is the number of parties. This overhead comes from the fact that we need to work over a larger structure (a Galois ring) in order to obtain algebraic properties that resemble those on fields, and that can be used for multiparty computation. A similar approach is taken in the SPD$\mathbb{Z}_{2^k}$ protocol [8] for computation over $\mathbb{Z}/2^k\mathbb{Z}$ by using the larger ring $\mathbb{Z}/2^{k+s}\mathbb{Z}$. In that work it is conjectured that this is an inherent price to pay for working over an algebraic structure with less nice properties than a field, and our current approach to information-theoretic MPC over $\mathbb{Z}/p^k\mathbb{Z}$ seems to support this claim, at least in the setting of a single circuit execution.

We consider as future work improving the complexity of the protocols presented here (specially the one from Sect. 4 for honest majority) by adapting more efficient protocols over fields like [5], whose complexity is almost-linear in the number of parties.

Acknowledgements. This work has been supported by the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No. 669255 (MPCPRO) and ERC ADG grant No. 74079 (ALGSTRONGCRYPTO).

References

1. Araki, T., et al.: Generalizing the SPDZ compiler for other protocols. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 880–895. ACM (2018)
2. Barak, A., Escudero, D., Dalskov, A., Keller, M.: Secure evaluation of quantized neural networks. Cryptology ePrint Archive, Report 2019/131 (2019). https://eprint.iacr.org/2019/131
3. Beerliová-Trubíniová, Z., Hirt, M.: Efficient multi-party computation with dispute control. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 305–328. Springer, Heidelberg (2006). https://doi.org/10.1007/11681878_16
4. Beerliová-Trubíniová, Z., Hirt, M.: Perfectly-secure MPC with linear communication complexity. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 213–230. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78524-8_13
5. Ben-Sasson, E., Fehr, S., Ostrovsky, R.: Near-linear unconditionally-secure multiparty computation with a dishonest minority. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 663–680. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_39
6. Cantor, D.G., Kaltofen, E.: On fast multiplication of polynomials over arbitrary algebras. Acta Inf. **28**(7), 693–701 (1991)
7. Cascudo, I., Cramer, R., Xing, C., Yuan, C.: Amortized complexity of information-theoretically secure MPC revisited. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10993, pp. 395–426. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96878-0_14
8. Cramer, R., Damgård, I., Escudero, D., Scholl, P., Xing, C.: SPD$\mathbb{Z}_{2^k}$: efficient MPC mod 2^k for dishonest majority. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10992, pp. 769–798. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_26
9. Cramer, R., Damgård, I., Nielsen, J.B.: Secure Multiparty Computation and Secret Sharing. Cambridge University Press, Cambridge (2015)
10. Cramer, R., Fehr, S., Ishai, Y., Kushilevitz, E.: Efficient multi-party computation over rings. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 596–613. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_37

11. 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. IACR Cryptology ePrint Archive 2019/832 (2019)
12. Damgard, I., Keller, M., Larraia, E., Pastro, V., Scholl, P., Smart, N.P.: Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. Cryptology ePrint Archive, Report 2012/642 (2012)
13. Damgård, I., Escudero, D., Frederiksen, T., Keller, M., Scholl, P., Volgushev, N.: New primitives for actively-secure MPC over rings with applications to private machine learning. In: 2019 2019 IEEE Symposium on Security and Privacy (SP), pp. 1325–1343. IEEE Computer Society, Los Alamitos (2019)
14. Keller, M., Orsini, E., Scholl, P.: MASCOT: faster malicious arithmetic secure computation with oblivious transfer. Cryptology ePrint Archive, Report 2016/505 (2016). http://eprint.iacr.org/2016/505
15. Massey, J.: Shift-register synthesis and BCH decoding. IEEE Trans. Inf. Theory **15**(1), 122–127 (1969)
16. Wan, Z.-X.: Lectures on Finite Fields and Galois Rings. World Scientific Publishing Company, Singapore (2003)