Asymptotically Good Multiplicative LSSS over Galois Rings and Applications to MPC over $\mathbb{Z}/p^k\mathbb{Z}$

Mark Abspoel[1], Ronald Cramer[1,2], Ivan Damgård[3], Daniel Escudero[3], Matthieu Rambaud[4], Chaoping Xing[5], and Chen Yuan[1]

[1] Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands
[2] Mathematisch Instituut, Leiden University, The Netherlands
[3] Aarhus University, Denmark
[4] Telecom Paris, Institut Polytechnique de Paris
[5] School of Electronic Information and Electric Engineering, Shanghai Jiaotong University, Shanghai, China

Abstract. We study information-theoretic multiparty computation (MPC) protocols over rings $\mathbb{Z}/p^k\mathbb{Z}$ that have good asymptotic communication complexity for a large number of players. An important ingredient for such protocols is arithmetic secret sharing, i.e., linear secret-sharing schemes with multiplicative properties. The standard way to obtain these over fields is with a family of linear codes C, such that C, $C^\perp$ and C^2 are asymptotically good (strongly multiplicative). For our purposes here it suffices if the square code C^2 is not the whole space, i.e., has codimension at least 1 (multiplicative).

Our approach is to lift such a family of codes defined over a finite field $\mathbb{F}$ to a Galois ring, which is a local ring that has $\mathbb{F}$ as its residue field and that contains $\mathbb{Z}/p^k\mathbb{Z}$ as a subring, and thus enables arithmetic that is compatible with both structures. Although arbitrary lifts preserve the distance and dual distance of a code, as we demonstrate with a counterexample, the multiplicative property is not preserved. We work around this issue by showing a dedicated lift that preserves *self-orthogonality* (as well as distance and dual distance), for $p \geq 3$. Self-orthogonal codes are multiplicative, therefore we can use existing results of asymptotically good self-dual codes over fields to obtain arithmetic secret sharing over Galois rings. For $p = 2$ we obtain multiplicativity by using existing techniques of secret-sharing using both C and $C^\perp$, incurring a constant overhead. As a result, we obtain asymptotically good arithmetic secret-sharing schemes over Galois rings.

With these schemes in hand, we extend existing field-based MPC protocols to obtain MPC over $\mathbb{Z}/p^k\mathbb{Z}$, in the setting of a submaximal adversary corrupting less than a fraction $1/2 - \varepsilon$ of the players, where $\varepsilon > 0$ is arbitrarily small. We consider 3 different corruption models. For passive and active security with abort, our protocols communicate $O(n)$ bits per multiplication. For full security with guaranteed output delivery we use a preprocessing model and get $O(n)$ bits per multiplication in the online phase and $O(n \log n)$ bits per multiplication in the offline phase. Thus, we obtain true linear bit complexities, without the common assumption that the ring size depends on the number of players.

1 Introduction

A secret-sharing scheme is a mathematical object that disperses a secret element into n shares. Combined, the shares determine the secret, but individual shares, and limited subsets of them, contain no information about the secret. In linear secret-sharing schemes (LSSS), given several secret-shared elements, linear operations on the secrets correspond to linear operations on the shares. LSSS are the cornerstone of information-theoretic multiparty computation (MPC) protocols, but they also have applications in other domains of cryptography.

LSSS and MPC are typically defined over finite fields (e.g., the secret and the shares are elements of the same finite field), which have a rich algebraic structure. A natural question is whether we can extend some of these techniques to other structures, such as rings $\mathbb{Z}/p^k\mathbb{Z}$ where $k > 0$ is an integer and p is a prime. This question is not only motivated in theory: some results [17,2] show that MPC over $\mathbb{Z}/2^k\mathbb{Z}$ with $k = 32$ or $k = 64$ can offer many practical benefits compared to fields, partly due to the compatibility of binary arithmetic in modern hardware. Feasibility of LSSS and MPC over these rings, as well as theoretical benefits, were already demonstrated back in 2003 based on *black-box secret sharing* [13].

Recently, MPC *directly* over these rings has been shown, in both the cryptographic dishonest majority setting [11] as well as the information-theoretic setting [1], by extending and generalizing existing techniques over fields. Both of these approaches use a single LSSS defined over $\mathbb{Z}/p^k\mathbb{Z}$: additive secret sharing and a variant of Shamir's secret sharing, respectively. It is natural to wonder whether techniques for other LSSS can be extended to these rings, to obtain desirable properties such as good asymptotic complexity.

In this work, we study the lifting of linear codes defined over finite fields to *Galois rings*, which are a natural generalization of *both* finite fields and our rings of interest $\mathbb{Z}/p^k\mathbb{Z}$. In a way, Galois rings are analogues of finite fields: informally, a Galois ring is to $\mathbb{Z}/p^k\mathbb{Z}$, what a finite field is to the prime field $\mathbb{F}_p$. Therefore, to extend existing techniques over finite fields to work over $\mathbb{Z}/p^k\mathbb{Z}$, it is necessary to consider Galois rings. As shown in Section 3, lifting preserves the essential properties of linear codes (distance, dual distance) that make them suitable as LSSS.

However, extending the theory of LSSS to Galois rings is not straightforward, due to the reduced structure and the presence of non-invertible elements. Thus, a priori it is not clear if properties of LSSS over fields carry over when considering these constructions over Galois rings. The above leads to the following question: *Can we obtain "good" LSSS over a Galois ring?* More precisely, we focus on realizing families of LSSS indexed by n, the number of shares, with privacy and reconstruction thresholds arbitrarily close to $n/2$, and with the information rate tending to a positive constant. The most widely known construction of LSSS over fields, Shamir secret sharing, does not satisfy the rate condition as it is based on polynomial interpolation and therefore the shares have to be at least $\log(n)$ in length. This issue was addressed over fields in the work of [9], using non-trivial results on random codes.

The above question is relevant for MPC that is asymptotically optimal, i.e., secure multiplication that has a total communication complexity linear in the number of players [15]. For information-theoretic MPC we typically care about *arithmetic secret-sharing schemes*, or synonymously, LSSS with multiplicativity: given two secret-shared elements, their product is a linear function of the pairwise products of shares. However, as we shall demonstrate with a counterexample in Section 3, multiplicativity is not straightforward to achieve over Galois rings.

True linear complexity is hard to achieve, and in fact conjectured to be impossible in the maximal adversary case $n = 2t + 1$ for the single-circuit setting.[6] Many state-of-the-art protocols such as [4,20] state a linear complexity, but the complexity is given in the number of field elements communicated. If the field is fixed and the number of players tends to infinity, this obscures a $\log(n)$ factor in the bit-complexity of the protocol. Over fields, this asymptotic factor does not affect the complexity for practical ranges of parameters, since the field size is usually much larger than the number of players. However, for our rings $\mathbb{Z}/p^k\mathbb{Z}$ this issue is more pressing, since the comparable requirement is that $p > n$ rather than $p^k > n$ [1], thus leading to a $\log(n)$ factor immediately if for example $p = 2$. Removing this $\log(n)$ overhead is thus worthwhile and in fact highly desired, since it would achieve a *constant* complexity per party: even if more parties join the computation, the communication per party does not increase.

1.1 Our Contributions

We show that some of the results for LSSS over a finite field $\mathbb{F}$ also hold over a Galois ring R that contains $\mathbb{F}$ as a residue field, by *arbitrarily* lifting the associated code over $\mathbb{F}$ to R, and showing that certain relevant properties are preserved.

First, in Section 3, we show that we can obtain explicit good families of linear codes over Galois rings. In what follows, R is a large enough Galois ring.

Theorem 1 (informal). *There exists an explicit family of R-linear codes over R with $|R| = O_\varepsilon(1)$ such that its relative distance is at least $\frac{1}{2} - \varepsilon$ and relative dual distance is at least $\frac{1}{2} - \varepsilon$. In particular, there exists an explicit family of self-dual codes over R with relative distance at least $\frac{1}{2} - \varepsilon$.*

It is well-known that any linear code over a field with good parameters yields a good linear secret-sharing scheme [24], and it is straightforward to show this also holds over Galois rings. However, to get the arithmetic secret-sharing schemes that we need, we also need good parameters for the square of the code. We demonstrate with a counterexample that these parameters are not preserved by arbitrary lifts.

We work around this issue by showing a *dedicated* lift for $p > 2$ that preserves self-orthogonality in Section 3.1. For $p = 2$ we secret-share elements using both

[6] LSSS with these parameters are equivalent to MDS codes, hence if the MDS conjecture if true, then the field size has to grow with the number of players. When evaluating a circuit multiple times in parallel, this can be mitigated [8].

C and $C^{\perp}$, at the expense of increasing the share size by a factor of two. Both of these approaches rely on techniques from [12] to obtain arithmetic secret sharing via a code and its dual, and we demonstrate in Section 4 that these extend to Galois rings. We capture the asymptotic result in the following theorem.

Theorem 2 (informal). *There exists a family of R-arithmetic secret-sharing schemes $\Sigma_1, \Sigma_2, \ldots$ over R with $|R| = O_\varepsilon(1)$ such that the number of players $n(\Sigma_i) \to \infty$, and the schemes have $t(\Sigma_i) \geq (1/2 - \varepsilon)n(\Sigma_i)$-privacy and $r(\Sigma_i) \geq (1/2 - \varepsilon)n(\Sigma_i)$-reconstruction.*

To illustrate the power of our results on arithmetic secret sharing, we apply them to the problem of communication-efficient honest-majority MPC over $\mathbb{Z}/p^k\mathbb{Z}$. This problem has only recently been studied in [1], but the authors were more concerned with feasibility rather than achieving optimal communication complexity. In particular, their protocol is based on the (no longer state-of-the-art) protocol of [3], which has $O(n^2 \log(n))$ complexity in the number of parties. Here, the $\log(n)$ factor comes from polynomial interpolation, as discussed above. Plugging in our LSSS we immediately remove this $\log(n)$ factor and obtain true quadratic complexity for the adversary regime of $t < (1/2 - \varepsilon)n$, analogously to the work of [9] over fields.

We further improve the complexity, for three different regimes:

1. (Section 5) For passive security, we present a protocol that obtains an amortized communication complexity of $O(n)$ bits per multiplication gate.
2. (Section 5.3) We discuss how to extend the protocol above to the setting of active security with abort, achieving an amortized communication complexity of $O(n)$ bits per multiplication gate.
3. (Section 6) For full active security with guaranteed output delivery, we obtain an amortized communication complexity of $O(n \log(n))$ bits for the offline phase and $O(n)$ bits for online phase. This solves the open problem from [1].

The last protocol is the most involved, since we adapt the protocol of [5] to work over Galois rings. Here we achieve linear complexity only in the online phase, as we still rely on polynomial interpolation to efficiently verify multiplication triples in the preprocessing phase. This matches the state-of-the-art over fields until the very recent result of [21]. However, since their protocol also uses the constructions of [5], our techniques can be combined with theirs to achieve linear complexity for the preprocessing phase.[7]

1.2 Overview of our Techniques

We mainly use elementary (arbitrary) liftings from codes C over a finite field $\mathbb{F}$ to a Galois ring that contains $\mathbb{F}$ as its residue field, and $\mathbb{Z}/p^k\mathbb{Z}$ as a subring. This way we leverage results from codes over fields directly. For example, since there exist explicit families of codes with asymptotically good distance and dual distance

[7] Ignoring terms that are sublinear in the circuit size.

over a finite field $\mathbb{F}$, we also obtain explicit families of codes with asymptotically good distance and dual distance over R.

Once we obtain arithmetic secret sharing over R we can use it to get MPC over $\mathbb{Z}/p^k\mathbb{Z}$. Our general template to obtain an MPC protocol is to first develop protocols over R itself, and since $\mathbb{Z}/p^k\mathbb{Z}$ is a subring of R, we can supply inputs in $\mathbb{Z}/p^k\mathbb{Z}$ and then evaluate a circuit over R to securely obtain the correct output in $\mathbb{Z}/p^k\mathbb{Z}$.[8][9]

Our passively secure protocol over R follows the template of [16], which consists of preprocessing so-called "double-sharings" and then using them to compute secure multiplications in the online phase. Since our construction of arithmetic secret sharing does not come directly from a code, we abstract the underlying technique to work on arbitrary arithmetic secret-sharing schemes. We do not have access to Vandermonde matrices over R directly, but we fix this by moving to an extension of Galois rings without amortized overhead using the "tensoring trick" from [8] together with the interpolation theorems from [1].

To get our actively secure protocol with abort, we make the simple but powerful observation that our protocol above is already actively secure up to additive attacks, i.e., the only attack that an adversary may carry out is to add a chosen value to the outputs of multiplications that is independent of the inputs. We obtain our actively secure protocol with abort by compiling our passively secure protocol with the recent work of [2], preserving linear complexity.[10]

Finally, for our actively secure protocol with guaranteed output delivery we use our arithmetic secret-sharing scheme as a building block and extend the protocol of [5], which is defined over a field in the $t < n/2$ regime. We show the check of authentication tags generalizes to our setting, and show how to compute the authentication tags (based on "twisted sharings") using our secret-sharing scheme. We also adapt the batch verification of triples.

1.3 Related Work

Honest majority MPC over rings has been already studied in [13] via black-box secret sharing, but their computational overhead is rather large. This problem was not revisited until very recently, with the work of [1], which presented efficient constructions using Galois rings, showing their potential benefits in the theory of MPC. They provide a protocol for multiplication with $O(n \log n)$ bits of total communication per gate, for the $t < n/3$ setting. This $\log(n)$ factor comes from using Shamir's scheme, and removing it requires codes with good distance

[8] One may think initially that R is more general than $\mathbb{Z}/p^k\mathbb{Z}$ and thefore computation over $\mathbb{Z}/p^k\mathbb{Z}$ is implied trivially by computation over R by taking the degree of the extension to be 1. However, note that the degree of the extension is constrained to be $\Omega(\log_p(\varepsilon^{-1}))$, which is constant for a fixed $\varepsilon > 0$, but it is not necessarily equal to 1.

[9] For passive security the condition on the inputs is trivial to satisfy, but for active security some extra check needs to be added, which was already addressed in [1] for the case of Galois rings.

[10] Although their compiler is described for $\mathbb{Z}/p^k\mathbb{Z}$, it also applies to arbitrary Galois rings.

of the square, or asymptotically good families of reverse multiplication-friendly embeddings, which as we illustrate in Example 2 are out of reach of our elementary lifting methods. Both were very recently claimed by [14], and illustrated with protocols for the $t < n/3$ setting. The tools developed in the present work enable up to honest majority, so are therefore complementary. Also, the recent work of [2] considers honest majority MPC over $\mathbb{Z}/2^k\mathbb{Z}$, but they achieve only security with abort and they do so with a communication complexity of $O(n\log(n))$ for both online and offline phases.

On the other hand, there are several other works in the context of honest-majority MPC over *fields*. We have already mentioned the work of Ben-Sasson et al. [5], that proposes a protocol in the honest-majority setting with guaranteed output delivery and near-linear communication complexity, and constitutes the basis of our protocol in Section 6. More recently, the protocol of [21] improves upon the protocol in [5] by introducing a novel method for verifying the correctness of multiplication triples. In the setting of security with abort the line of research is richer, with many protocols proposed in the last few years that aim at providing concrete practical efficiency. For example, an efficient general compiler from active security up to additive attacks to active security is presented in [10], which improves upon the methods built in [23]. The work of [25] also improves upon [23] by extending it using similar ideas as the batch triple check presented in [5]. Also, very recently, an efficient method to achieve actively secure three party computation was presented [7], building on top of the distributed zero knowledge proof techniques introduced in [6]. Although the authors of this work do consider an extension of their protocol to the ring $\mathbb{Z}/2^k\mathbb{Z}$, this technique is unlikely to be efficient in practice as it uses a Galois ring of a degree that is roughly equal to the security parameter.

2 Preliminaries

2.1 Linear codes over finite fields

Let $\mathbb{F}_q$ be the finite field with q elements, and let $\mathbb{F}_q^n$ be the $\mathbb{F}_q$-vector space consisting of n copies of $\mathbb{F}_q$. A code $C \subseteq \mathbb{F}_q^n$ is a set of row vectors in $\mathbb{F}_q^n$. The rate of C is defined as $\frac{\log_q |C|}{n}$. For a (row) vector $\mathbf{x} = (x_1, \ldots, x_n)$ its Hamming weight is the number of nonzero coordinates: $w_H(\mathbf{x}) = |\{i \in [n] \mid x_i \neq 0\}|$, where we write $[n] := \{1, \ldots, n\}$. If $\mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{F}_q^n$ is another vector, the Hamming distance between $\mathbf{x}$ and $\mathbf{y}$ is the number of coordinates in which they differ $d(\mathbf{x}, \mathbf{y}) = |\{i \in [n] \mid x_i \neq y_i\}| = w_H(\mathbf{x} - \mathbf{y})$. The minimum distance of a code C is defined as $d(C) = \min_{\mathbf{x} \neq \mathbf{y} \in C} d(\mathbf{x}, \mathbf{y})$.

In the following, let $C \subseteq \mathbb{F}_q^n$ be a linear subspace; we then say C is a linear code. The dimension of C is the dimension of C as a vector space. If C has $k = \dim(C)$ and $d = d(C)$, we say C is a $[n, k, d]$-linear code over $\mathbb{F}_q$. A matrix G is a generator matrix for C if its rows form a basis for C. The dual code of C is defined as $C^\perp = \{\mathbf{x} \in \mathbb{F}_q^n \mid \forall \mathbf{y} \in C : \mathbf{x}\mathbf{y}^T = 0\}$. One can see that $C^\perp$ is a linear code with dimension $n - \dim_{\mathbb{F}_q}(C)$. The dual distance of C is defined as the minimum distance of $C^\perp$, and is denoted as $d^\perp(C)$.

In this paper, we are mostly concerned with the minimum distance d and dual distance $d^\perp$ of a linear code C. For applications to secret sharing, we want both of these to be large, since they imply $(n-d+1)$-reconstruction and $(d^\perp-1)$-privacy for secret-sharing scheme associated to the code. There is a large body of works dedicated to determining the achievable distance and dual distance of a code. In this work, we are particularly interested in the asymptotic behavior of d and $d^\perp$. To characterize this asymptotic behavior, we look at the relative distance $\delta = \frac{d}{n}$ and relative dual distance $\delta^\perp = \frac{d^\perp}{n}$.

Definition 1. *A family $C_1, C_2, \ldots$ of linear codes over a fixed finite field, where each C_i has parameters $[n_i, k_i, d_i]$ and dual distance $d_i^\perp$, is said to have relative distance δ and relative dual distance $\delta^\perp$ if the following holds:*

1. $\lim_{i\to\infty} n_i = \infty$
2. $\liminf_{i\to\infty} \frac{d_i}{n_i} \geq \delta, \quad \liminf_{i\to\infty} \frac{d_i^\perp}{n_i} \geq \delta^\perp.$

We stress that we study this asymptotic behaviour only for a family of codes defined over the same finite field.

In general, there are two ways to construct a family of codes with large relative distance and relative dual distance. One way is through a random argument that gives a family of codes reaching the Gilbert-Varshamov bound. For a finite field $\mathbb{F}_q$ with $q < 49$, this Gilbert-Varshamov Bound is the best lower bound known. When $q \geq 49$ is a square, there exists an explicit construction of algebraic geometric codes outperforming the random codes, i.e., there exists a family of algebraic geometric codes attaining the celebrated Vlăduţ-Drinfeld bound [18]. We skip the details of these codes and refer the interested reader to [27]. The family of algebraic geometric codes attaining the celebrated Vlăduţ-Drinfeld bound meets the following condition.

Proposition 1. *Let q be any prime power. Then there exists an explicit family of codes over a fixed finite field $\mathbb{F}_{q^2}$ with relative distance δ and relative dual distance $\delta^\perp$ as long as δ and $\delta^\perp$ satisfy*

$$\delta + \delta^\perp \leq 1 - \frac{2}{q-1}. \tag{1}$$

A similar result holds for self-dual codes [26], i.e., there exists an explicit family of self-dual codes reaching the Vlăduţ-Drinfeld bound.

Proposition 2. *Let $\varepsilon > 0$ be any small constant. Then, for any $q \geq 2/\varepsilon$ there exists an explicit family of codes over a fixed finite field $\mathbb{F}_{q^2}$ such that its relative distance $\delta \geq \frac{1}{2} - \varepsilon$ and its relative dual distance $\delta^\perp \geq \frac{1}{2} - \varepsilon$. Moreover, there exists an explicit family of self-dual codes over a fixed finite field $\mathbb{F}_{q^2}$ with relative distance $\delta \geq \frac{1}{2} - \varepsilon$.*

2.2 Galois Rings

Galois rings are a natural analogue to finite fields: roughly, Galois rings are to $\mathbb{Z}/p^k\mathbb{Z}$ what finite fields are to prime-order fields $\mathbb{F}_p$. As such, these rings have rich structure and they share many properties with finite fields. In fact, Galois rings are a strict generalization of finite fields, since setting $k = 1$ one obtains exactly the finite fields.

Definition 2. *Let p be a prime number and let k be a positive integer. Let $g(Y) \in (\mathbb{Z}/p^k\mathbb{Z})[Y]$ be a monic polynomial such that its reduction modulo p is an irreducible polynomial in $\mathbb{F}_p[Y]$. The ring*

$$R := (\mathbb{Z}/p^k\mathbb{Z})[Y]/\left(g(Y)\right)$$

is called a Galois ring.

Proposition 3. *R has the following properties:*

1. *It is a local ring, i.e. it has a unique maximal ideal $(p) \subsetneq R$. We have that $R/(p) \cong \mathbb{F} := \mathbb{F}_{p^h}$, where h denotes the degree of g.*
2. *The Lenstra constant of R is p^h, which gives the maximum number of interpolation points in Shamir's (because the pairwise differences must be invertible)*
3. *For any prime p, positive integer k, and positive integer h there exists a Galois ring as defined above, and any two of them with identical parameters p, k, h are isomorphic. We may therefore write $R = \mathsf{GR}(p^k, h)$.*
4. *If e is any positive integer, then R is a subring of $\hat{R} = \mathsf{GR}(p^k, h \cdot e)$. There is a polynomial $\hat{g} \in R[X]$ that is irreducible modulo p, such that $\hat{R} = R[X]/(\hat{g}(X))$. There is a natural R-module isomorphism $R^e \to \hat{R}$.*

Remark 1. Also, we have a natural ring embedding $\mathbb{Z}/p^k\mathbb{Z} \hookrightarrow R$, given by mapping $x \mapsto x \bmod g(Y)$. Moreover, there is another way to uniquely represent the elements of R. Since $R/(p) \cong \mathbb{F}$, let ξ be a non-zero element of order $p^h - 1$ in R and define the subset

$$\mathcal{I} = \{0, 1, \xi, \dots, \xi^{p^h-2}\} \subset R\,. \tag{2}$$

Then, any element $a \in R$ can be uniquely written as

$$a = a_0 + a_1 p + a_2 p^2 + \cdots + a_{k-1}p^{k-1} \text{ where } a_0, \dots, a_{k-1} \in \mathcal{I}\,.$$

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 $a/p^u := a_u + a_{u+1}p + \cdots + a_{k-1}p^{k-u-1} \in \mathsf{GR}(p^{k-u}, h)$; notice that $a/p^u \equiv a_u \pmod{p}$. If u is maximal and a is non-zero in R, then $a/p^u \in R^*$.

Finally, Item 1 of Proposition 3 gives rise to the canonical map $\pi : R \to \mathbb{F}$ ("reduction modulo p"), which we shall frequently use. It is easy to see that $\pi|_{\mathcal{I}}$ is a bijection, and in particular we have a one-to-one correspondence between $\mathcal{I}$ and $\mathbb{F}_{p^h}$. Given $x \in R$ we shall also write $\overline{x} = \pi(x)$.

2.3 Arithmetic Secret Sharing Schemes

Our definition of an arithmetic secret sharing scheme over a Galois ring is inspired by the notion a a general linear secret-sharing scheme from [8]. Let U be an R-algebra, $[n]$ be a finite set, and Z be an R-algebra. Suppose we have an R-submodule $C \subseteq U^n$ and a surjective R-module homomorphism $\psi : C \to Z$. For any subset $A \subseteq [n]$ we have projection homomorphisms $\pi_A : U^n \to U^{|A|}$ onto the coordinates indexed by A. For $i \in [n]$ we write $\pi_i := \pi_{\{i\}}$.

Let $A \subseteq [n]$ be a subset. Then we say:

1. A is a *reconstructing set* for ψ if $\ker \pi_A|_C \subseteq \ker \psi$.
2. A is a *privacy set* for ψ if the product of morphisms $\langle \pi_A|_C, \psi \rangle : C \to \pi_A(C) \times Z$ is surjective.

A is a reconstructing set for ψ iff any set of A-coordinates uniquely determines the corresponding value in Z. Also, A is a privacy set for ψ if given $\mathbf{x} \in C$, the A-coordinates $\pi_A(\mathbf{x})$ are independent of $\psi(\mathbf{x})$.

Definition 3. *An R-arithmetic secret-sharing scheme (or: R-ASSS) Σ consists of the following data:*

- *A number of players n.*
- *A finite commutative R-algebra U, the share space.*
- *A finite commutative R-algebra Z, the secret space.*
- *An R-submodule $C \subseteq U^n$, the defining module.*
- *The defining map: a surjective R-module homomorphism $\psi : C \to Z$ such that $[n]$ is a reconstructing set for ψ.*

For $\mathbf{x} = (x_1, \ldots, x_n), \mathbf{y} = (y_1, \ldots, y_n) \in U^n$, define $\mathbf{x} * \mathbf{y}$ as the coordinatewise product $(x_1 y_1, \ldots, x_n y_n)$. We require that the following criteria are satisfied:

1. *$[n]$ is a reconstructing set for (C, ψ).*
2. *There is a unique surjective R-module homomorphism $\overline{\psi} : C^{*2} \to Z$ such that $\overline{\psi}(x * y) = \psi(x)\psi(y)$, and such that $[n]$ is a reconstructing set for $\overline{\psi}$.*

We may refer to Σ as the 5-tuple (n, U, Z, C, ψ). We also say Σ is an R-arithmetic secret-sharing scheme of Z over U *with n players.*

We define the following terminology:

- Σ has t-privacy if all subsets $A \subseteq [n]$ of cardinality $|A| = t$ are privacy sets for ψ.
- Σ has r-reconstruction if all subsets $A \subseteq [n]$ of cardinality $|A| = r$ are reconstructing sets for ψ.

3 Codes over Galois Rings

We now show how to obtain codes over Galois rings; although there is a large body of works dedicated to linear codes, most of it only deals with codes over finite fields. For the purpose of asymptotically good secret-sharing schemes, we need a family of codes over Galois rings whose rate and relative distance tends to a positive constant.

We obtain such codes by *arbitrarily* lifting linear codes defined over some finite field $\mathbb{F}$, such as the ones from Proposition 1, to a Galois ring whose residue field is $\mathbb{F}$. We show that the lifted codes have at least the same distance and dual distance as the original codes, hence using Proposition 1 we obtain a good family of codes over Galois rings of arbitrary characteristic p^k.

For the particular case of self-orthogonal codes defined over a field of characteristic not equal to 2, we give an *explicit* lift that preserves self-orthogonality in Section 3.1. Self-orthogonal codes satisfy a multiplicative property that is needed for arithmetic secret sharing. In Section 4 we show how to extend existing techniques to obtain multiplication for $p = 2$, but this comes at the cost of doubling the share size.

Let $R = \mathsf{GR}(p^k, h)$ be a Galois ring with residue field $\mathbb{F} = \mathbb{F}_{p^h}$. We define a linear code C of length n over R to be a free R-submodule of R^n. We define its dimension as $\dim(C) = \operatorname{rank}_R(C)$. Recall the canonical homomorphism $\pi : R \to \mathbb{F}$. For convenience we will also write π for the induced map on vectors or matrices defined over R, and write $\overline{\mathbf{x}} := \pi(\mathbf{x})$ for $\mathbf{x} \in R^n$ and $\overline{M} = \pi(M)$ for a matrix M over R.

Proposition 4. *Let C be a linear code over R. Then the following statements hold:*

1. $\operatorname{rank}_R C = \dim_{\mathbb{F}} \overline{C}$*, where* $\overline{C} = \pi(C) \subseteq \mathbb{F}^n$ *is the reduction of* C *modulo* p.
2. *If* $\mathbf{c} \neq \mathbf{0} \in C$ *we may write* $\mathbf{c} = p^m\mathbf{y}$, *for* $0 \leq m < k$ *and* $\pi(\mathbf{y}) \neq \mathbf{0} \in \overline{C}$.

Proof. Let us prove the first claim. Since $C \subseteq R^n$ is a linear code, it has an R-basis $\mathbf{e}_1, \ldots, \mathbf{e}_t \in R^n$. Then, it is clear that $\overline{C}$ is an $\mathbb{F}$-linear code spanned by $\pi(\mathbf{e}_1), \ldots, \pi(\mathbf{e}_t)$. If we can show that $\pi(\mathbf{e}_1), \ldots, \pi(\mathbf{e}_t)$ are linearly independent over $\mathbb{F}$, then we are done. Assume this is false, so there exist $\lambda_1, \ldots, \lambda_k \in \mathbb{F}$ not all equal to 0 such that $\sum_{i=1}^t \lambda_i \pi(\mathbf{e}_i) = 0$. Let $\lambda'_i = \pi^{-1}(\lambda_i) \in \mathcal{I} \subseteq R$, then it holds that $\sum_{i=1}^t \lambda'_i \mathbf{e}_i \in pR$, since

$$\pi\left(\sum_{i=1}^t \lambda'_i \mathbf{e}_i\right) = \sum_{i=1}^t \lambda_i \pi(\mathbf{e}_i) = 0.$$

It follows that $\sum_{i=1}^t p^{k-1}\lambda'_i\mathbf{e}_i = 0$ and $p^{k-1}\lambda'_1, \ldots, p^{k-1}\lambda'_t$ are not all zero. This contradicts the claim that $\mathbf{e}_1, \ldots, \mathbf{e}_t$ form a basis of C.

We turn to the second claim. Let G be a $t \times n$ matrix over R whose rows form a basis $\mathbf{e}_1, \ldots, \mathbf{e}_t$ of C. We may represent $C = \{\mathbf{x}G : \mathbf{x} \in R^t\}$. We call G the generator matrix of C, which gives a linear isomorphism between R^t and

C. Let $\mathbf{c} = \mathbf{x}G$ be any nonzero codeword in C. Since G is an isomorphism, $\mathbf{x}$ is also a nonzero vector. By Remark 1, we write $\mathbf{x} = p^m\mathbf{x}_1$ with $0 \le m < k$ and $\mathbf{x}_1 \neq \mathbf{0} \in \mathcal{I}^t$. This follows that $\mathbf{c} = p^m\mathbf{x}_1 G$. Let $\mathbf{y} = \mathbf{x}_1 G$ and the desired result follows as $\pi(\mathbf{y}) = \pi(\mathbf{x}_1)\pi(G) \in \overline{C}$ is a nonzero codeword. □

Lemma 1. *Let $C \subseteq R^n$ be a linear code. We have $d(C) \geq d(\overline{C})$.*

Proof. Let G be the generator matrix of C. Since C is a linear code, it suffices to bound the weight of its codewords. For any $\mathbf{c} \neq \mathbf{0} \in C$, by Proposition 4 we can write $\mathbf{c} = p^m\mathbf{y}$ for some $\overline{\mathbf{y}} \neq \mathbf{0} \in \overline{C}$ and $m < k$. Note that $\overline{\mathbf{y}}$ is a nonzero codeword of $\overline{C}$. Thus, $w_H(\mathbf{c}) \geq w_H(\mathbf{y}) \geq d(\overline{C})$. The proof is completed. □

Example 1. To control the minimum distance, we need that C is a free module. Consider the code $\overline{C} := \langle(1,1,\dots,1)\rangle$ of two elements code over $(\mathbb{F}_2)^n$, with distance n. We can lift the code to $\mathbb{Z}/2^2\mathbb{Z}$ as $C := \langle(1,1,\dots,1),(2,0,\dots,0)\rangle$ which is non-free, because of the bad element $(2,0,\dots,0)$. Then $d(C) = 1 \ll d(\overline{C}) = n$.

Like for codes over a field, we can similarly define the dual code over R. The dual code of C is defined as $C^\perp = \{\mathbf{c} \in R^n \mid \mathbf{c}\mathbf{y}^T = \mathbf{0} \text{ for all } \mathbf{y} \in C\}$.

Lemma 2. *Assume that $C \subseteq R^n$ is a t-dimensional R-linear code. Then, $C^\perp \subseteq R^n$ is a $(n-t)$-dimensional R-linear code. Moreover, the minimum distance of $C^\perp$ is lower bounded by the minimum distance of the dual code of $\overline{C}$.*

Proof. Let G be the generator matrix of C. Every element in $C^\perp$ is a solution to the linear equation $G\mathbf{x}^T = \mathbf{0}$ over R, and vice versa. This implies that $C^\perp$ is the kernel $\ker(G)$ of the R-linear map $G\mathbf{x}^T$. The image $\mathrm{im}(G)$ of $G\mathbf{x}^T$ is C, a free module of R^n with rank t. The homomorphism theorem of modules states that $R^n/\ker(G) \cong \mathrm{im}(G)$. Thus, the kernel is also free and has rank $n-t$. By our definition, $\ker(G)$ is a linear code of dimension $n-t$ over R.

It remains to lower bound the minimum distance of $C^\perp$. Given any codeword $\mathbf{c} \neq \mathbf{0} \in C^\perp$, we have $G\mathbf{c}^T = \mathbf{0}$. Moreover, by Remark 1, we can write $\mathbf{c} = p^m\mathbf{y}$ for $0 \le m < k$ and $\mathbf{y} \neq \mathbf{0} \in \mathcal{I}^t$. Reducing modulo p gives $\overline{G}\overline{\mathbf{y}}^T = 0$ over $\mathbb{F}_{p^h}$. This implies that $\overline{\mathbf{y}}$ is a nonzero codeword in the dual code of $\overline{C}$. Then, the desired result follows. □

We now define the square of a linear code C over R. Given $\mathbf{x} = (x_1,\dots,x_n)$, $\mathbf{y} = (y_1,\dots,y_n) \in R^n$ denote their componentwise (Schur) product as $\mathbf{x} * \mathbf{y} = (x_1y_1,\dots,x_ny_n) \in R^n$. The square code C^{*2} is defined as $\mathrm{span}_R\{\mathbf{x} * \mathbf{y} \in R^n \mid \mathbf{x},\mathbf{y} \in C\}$. We emphasize that this square code C^{*2} is an R-module but not necessarily a free R-module. We say C is t-strongly multiplicative if the minimum distance $d(C)$, its dual distance $\mathrm{d}^\perp(C)$ and the distance of the square $d(C^{*2})$ are at least t.

One may wonder whether strong multiplication is preserved when lifting. Unfortunately, our next example shows that we can have poor distance of the square code C^{*2} even if $\overline{C}^{*2}$ is a square code with large distance.

Example 2. Let C_1 and C_2 be linear code over $\mathbb{F}_{p^h}$ such that C_1^{*2} and C_2^{*2} have distance d_1 and d_2 respectively. Let $S = GR(p^3, h)$ and C be a code over S defined as $C = \left\{ \left(\pi^{-1}(\mathbf{c}_1), p\,\pi^{-1}(\mathbf{c}_2)\right) \mid \mathbf{c}_1 \in C_1, \mathbf{c}_2 \in C_2 \right\}$. It is clear that $\overline{C} = \{(\mathbf{c}_1, 0) \mid \mathbf{c}_1 \in C_1\}$ whose square code has minimum distance d_1. On the other hand, since C_2^{*2} has distance d_2, let $\mathbf{y}_2 \in C_2^{*2}$ be a codeword with weight d_2. Then, we have that $(\mathbf{0}, p^2\pi(\mathbf{y}_2)) \in C^{*2}$, and therefore the minimum distance of C^{*2} is at most d_2. The desired result follows if we pick d_2 to be a small number and d_1 to a big number.

Unlike the distance and dual distance of the lifted code, strong multiplication does not automatically carry over. We now give a brief argument for uniformity, which shall be important when using our codes for secret sharing later on.

Lemma 3. *Let $C \subseteq R^n$ be a submodule, and let $U \subseteq [n]$ be an index set with $|U| \leq d(C^\perp) - 1$. Then the projection C_U of C onto the coordinates of U equals the whole space $R^{|U|}$.*

Proof. We argue by contradiction. Note that C_U is also an R-module, so we may write $C_U = \sum_{i=1}^t R\mathbf{x}_i$ with $t \leq |U|$. Here, C_U may be non-free. Let M be an $t \times |U|$ matrix whose rows are $\mathbf{x}_1, \ldots, \mathbf{x}_t$. Recall $\overline{M} = \pi(M)$ is the reduction of M modulo p.

We first show that if $|C_U| < R^{|U|}$, then the rank of $\overline{M}$ is less than $|U|$. It is obvious if $t < |U|$. When $t = |U|$, since $|C_U| < R^{|U|}$, $\mathbf{x}_1, \ldots, \mathbf{x}_t$ are linearly dependent over R. Therefore, there exist $\lambda_1, \ldots, \lambda_t \in R$, not all equal to 0, such that $\sum_{i=1}^t \lambda_i \mathbf{x}_i = 0$. Let m be maximal such that p^m divides all of $\lambda_1, \ldots, \lambda_t$. Then, we have $\sum_{i=1}^t \frac{\lambda_i}{p^m}\mathbf{x}_i = 0$. This implies that $\sum_{i=1}^t \pi(\frac{\lambda_i}{p^m})\pi(\mathbf{x}_i) = 0$ over $\mathbb{F}$ where $\pi(\frac{\lambda_1}{p^m}), \ldots, \pi(\frac{\lambda_t}{p^m}) \in \mathbb{F}$ are not all zero. Therefore, $\pi(\mathbf{x}_1), \ldots, \pi(\mathbf{x}_t)$ are linearly dependent and the rank of $\overline{M}$ is less than $|U|$.

Let $\mathbf{c}^T$ be the nonzero solution to $\overline{M}\mathbf{c}^T = \mathbf{0}$ over $\mathbb{F}$. Then, we have $Mp^{k-1}\pi^{-1}(\mathbf{c})^T = \mathbf{0}$ over R. Extend $p^{k-1}\pi^{-1}(\mathbf{c})^T$ to a vector $\mathbf{c}'$ in R^n by setting i-th component with $i \notin U$ to be zero. Clearly, $\mathbf{c}'$ is a codeword of the dual code $C^\perp$. However, $w_H(\mathbf{c}') = w_H(\mathbf{c}) \leq |U|$ and a contradiction occurs. □

3.1 Constructing A Self-Orthogonal Code over R

By a judicious choice of lift, we show that for $p \geq 3$ we can preserve self-orthogonality of a code over $\mathbb{F}$ when lifting to R.

Theorem 3. *Assume that there is a $[n, t, d]$ self-orthogonal code C over the finite field $\mathbb{F}_{p^h}$ with dual distance $d^\perp$ and $p \geq 3$. Then, there is a $[n, t, d]$ self-orthogonal code C_k with dual distance $d^\perp$ over the Galois ring $\mathsf{GR}(p^k, h)$ for any positive integer k. Moreover, given an explicit generator matrix of C the generator matrix of C_k is explicit.*

Proof. We lift the self-orthogonal code C increasing k step by step. For each step, we specify the lifted code by its generator matrix. Define $R_k := \mathsf{GR}(p^k, h)$. By Definition 2, R_k contains $\mathbb{Z}/p^k\mathbb{Z}$ as a subring, and its residue field is $\mathbb{F}_{p^h}$.

Our first step is to lift self-orthogonal code from $\mathbb{F}_{p^h}$ to $R_2 = \mathsf{GR}(p^2, h)$. Let C be an $[n, t, d]$ self-orthogonal code over $\mathbb{F}_{p^h}$ and $\overline{G} = \begin{pmatrix} \overline{I} & \overline{A} \end{pmatrix}$ [11] be the generator matrix of C. Due to the bijection between $\mathcal{I}$ and $\mathbb{F}_{p^h}$, we can find a matrix $G = \begin{pmatrix} I & A \end{pmatrix}$ with $G \in \pi^{-1}(\overline{G})$ whose entries are in $\mathcal{I}$. Self-orthogonality of C implies that

$$GG^T = AA^T + I = \overline{A} \times \overline{A}^T + \overline{I} = 0 \pmod{p}.$$

That means that all the entries in GG^T are elements in the ideal pR. By Remark 1, we can find a matrix S_1 over $\mathcal{I}$ such that $I + AA^T = pS_1 \pmod{p^2}$. It is clear that we can choose S_1 to be symmetric. Note that 2 is a unit in R_k as $p \neq 2$ and we can define $A_1 = A + 2^{-1}pS_1A$. Let $G_1 = \begin{pmatrix} I & A_1 \end{pmatrix}$ and let C_1 be the code whose generator matrix is G_1. Obviously, G_1 is defined over R_2. Next, we show that C_1 is indeed a self-orthogonal code over R_2. To see this, we have

$$\begin{aligned} G_1G_1^T &= I + AA^T + \frac{p}{2}(S_1AA^T + AA^TS_1) \\ &= pS_1 + \frac{p}{2}S_1(pS_1 - I) + \frac{p}{2}(pS_1 - I)S_1 \\ &= pS_1 - pS_1 = 0 \pmod{p^2}. \end{aligned}$$

The first equality follows from the fact that S_1 is a symmetric matrix. It remains to bound the minimum distance of C_1. Observe that the reduced code of C_1 is C. By Lemma 1, the distance of C_1 is lower bounded by that of C. We can apply the same argument to its dual distance by observing that the generator matrix of $C_1^\perp$ is $\begin{pmatrix} -A_1^T & I \end{pmatrix}$, whose reduction modulo p, the matrix $\begin{pmatrix} -A^T & I \end{pmatrix}$, is the generator matrix of $C^\perp$, and therefore C_1 is free. Now, C_2 is a self-orthogonal code over R_2 satisfying all the claims in our theorem. In the same manner, we can the lift code C_2 to a code C_3 over R_3. By induction, we obtain a code C_k over R_k for any $k \geq 1$ satisfying all the claims in our theorem. □

Note that a self-dual code is also a self-orthogonal code. Theorem 3 together with Proposition 2 gives the following.

Corollary 1. *Let $\varepsilon > 0$ be any small constant, k any positive integer and $p^h \geq \frac{4}{\varepsilon^2}$ be any square with p an odd prime. Then there exists an explicit family of self-dual codes over Galois ring $\mathsf{GR}(p^k, h)$ with relative distance $\delta \geq \frac{1}{2} - \varepsilon$.*

3.2 Code and Dual Code over R

Self-orthogonal codes satisfy a multiplicative property that immediately gives arithmetic secret sharing, as we will demonstrate in Section 4. However, we are only able to preserve self-orthogonality when $p \geq 3$, and the case $p = 2$ is also very interesting, especially for the purposes of MPC. The existence of asymptotically good self-orthogonal codes over these rings is not yet known. To get around this obstacle for $p = 2$, we will secret-share elements using both the code C and its

[11] We use bar notation to represent the fact that these matrices are defined over $\mathbb{F}_{p^h}$.

dual $C^{\perp}$ at the same time. This will double the share size, and hence double the communication complexity of the protocols we build on top of it. Since this factor 2 is constant, it does not affect our asymptotic complexity.

The following shows we can simulatenously lift a code and its dual to R and maintain distance and dual distance.

Theorem 4. *Assume that there is a $[n, t, d]$ linear code C over the finite field $\mathbb{F}_{p^h}$ with dual distance $d^{\perp}$. Then, there is a $[n, t, d]$ linear code C_k with dual distance $d^{\perp}$ over the Galois ring $\mathsf{GR}(p^k, h)$, for any integer k. Moreover, the generator matrices of C_k and its dual code are explicit as long as the generator matrix of C is explicit.*

Proof. Let G and H be the generator matrix and parity check matrix, respectively, of C. Note that H is also the generator matrix of $C^{\perp}$, the dual code of C over $\mathbb{F}_{p^h}$. We have $GH^T = 0 \pmod{p}$ and thus $GH^T = pM \pmod{p^2}$, for some matrix M defined over $\mathbb{F}_{p^h}$. Since G is a generator matrix of C, its rank is t. There exists $(n-t) \times n$ matrix A_1 such that $GA_1^T = -M \pmod{p}$. It follows that $G(H + pA_1)^T = GH^T + pGA_1^T = 0 \pmod{p^2}$.

Let C_2 be the linear code over $\mathsf{GR}(p^2, h)$ with generator matrix G. We claim that the dual code $C_2^{\perp}$ of C_2 has generator matrix $H + pA_1$. By Lemma 2, the dual code $C_2^{\perp}$ has dimension $n - t$. To see this, we first note that $H + pA_1$ has rank $\mathrm{rank}_{\mathbb{F}_{p^h}}(H) = n - t$ due to Proposition 4. Moreover, any codeword generated by $H + pA_1$ is a solution to $G\mathbf{x} = 0$ over R_2 since $G(H + pA_1)^T = 0 \pmod{p^2}$.

These two facts lead to the conclusion that $H + pA_1$ is indeed the generator matrix of $C_2^{\perp}$ over $\mathsf{GR}(p^2, h)$. The distance and dual distance follows from Lemma 1 and Lemma 2. In the same manner, one can show that C_k is a linear code over $\mathsf{GR}(p^k, h)$ with generator matrix G for any $k \geq 1$. In the meantime, by Lemma 1 the minimum distance and dual distance of C_k are lower bounded by d and $d^{\perp}$ respectively. The dual code of C_k is specified by its generator matrix $H + pA_1 + \cdots + p^{k-1}A_{k-1}$. □

Theorem 4 combined with Proposition 2 gives the following result.

Theorem 5. *Let $\varepsilon > 0$ be any small constant and $p^h \geq \frac{4}{\varepsilon^2}$ be any square. There exists an explicit family of codes over the Galois ring $\mathsf{GR}(p^k, h)$ with relative distance $\delta \geq \frac{1}{2} - \varepsilon$ and relative dual distance $\delta^{\perp} \geq \frac{1}{2} - \varepsilon$ for any integer k.*

4 Arithmetic Secret-Sharing over Galois Rings

In this section we construct an arithmetic secret-sharing scheme over a Galois ring R starting from an R-linear code C together with its dual $C^{\perp}$, by extending techniques from [12]. In this section, let $R = \mathsf{GR}(p^k, h)$, and suppose $C \subseteq R^{n+1}$ is a linear code with distance d and dual distance $d^{\perp}$. We first provide a brief overview of the techniques, before fixing the slightly heavier notation in Section 4.1 that we use to write the protocols in the remaining sections of this paper.

As for nomenclature, note that the difference between arithmetic and linear secret sharing is that the former is an LSSS with multiplication. We say an

LSSS has multiplication if there exists a multiplication operator $*$ on shares, such that given secrets x and y with respective share vectors $(x_1, \dots, x_n)$ and $(y_1, \dots, y_n)$ then the product $x \cdot y$ is linearly determined by the $*$-products of shares $x_1 * y_1, \dots, x_n * y_n$. Here we are explicit about the operator $*$ because in the arithmetic secret-sharing scheme that we construct, the shares are not elements of R, but rather each share is given by 2 elements in R. These pairs of R-elements form an R-algebra with the operator $*$, which we define below.

Recall that a secret-sharing scheme has t-privacy if for any share vector, any t coordinates are independent of the secret, and it has r-reconstruction if any r coordinates of a share vector jointly determine the secret. Via Massey's construction [24] we may obtain an LSSS from a code over a field with good parameters, and this generalizes to Galois rings, as follows. To share $s \in R$, we sample a codeword $\mathbf{c} = (s, c_1, \dots, c_n) \in C$ uniformly at random and let c_i be the i-th share. Due to properties of the dual distance $d^\perp$, we can show that for any subset $T \subseteq [n]$ with $|T| \leq d^\perp - 2$ and $s \in R$, $\{(c_i)_{i \in T} : (s, c_1, \dots, c_n) \in C\} = R^{|T|}$. This implies $(d^\perp - 2)$-privacy. From the minimum distance of C it follows that the LSSS has $(n - d + 1)$-reconstruction.

To use a secret-sharing scheme for MPC, we need the multiplicative property. The LSSS constructed above has multiplication if and only if its square code C^{*2} has minimum distance $d(C^{*2}) \geq 1$. Unfortunately, the codes from Theorem 5 do not satisfy this property. However, by simultaneously secret-sharing values in C and in the dual code $C^\perp$, we can obtain multiplication with the following construction from [12].

To secret-share $s \in R$, we sample a codeword $\mathbf{x} = (s, x_1, \dots, x_n) \in C$ and a codeword $\mathbf{y} = (s, y_1, \dots, y_n) \in C^\perp$ uniformly at random. The i-th share is now a pair (x_i, y_i). The privacy of this scheme is $\min\{d - 2, d^\perp - 2\}$ and it is $\min\{n - d + 1, n - d^\perp + 1\}$-reconstruction. Now suppose we have another secret-shared element $u \in R$ shared as $\mathbf{x}' = (u, x'_1, \dots, x'_n) \in C$ and $\mathbf{y}' = (u, y'_1, \dots, y'_n) \in C^\perp$. For the product su, we see that $\sum_{i=1}^n x_i y'_i = -su$ (and also $\sum_{i=1}^n y_i x'_i = -su$). This can be used to construct a linear secret-sharing scheme with multiplication, as we describe next.

4.1 Formalization

We now formalize the scheme and define the notation which we shall use in the remaining sections. Recall C is of length $n+1$. Let $\widetilde{C} \subseteq R^n$ denote the projection of C onto its last n coordinates, and similarly for $\widetilde{C^\perp} \subseteq R^n$. Let $\psi : \widetilde{C} \to R$ be the R-module homomorphism given by $\psi(x_1, \dots, x_n) = x$ where $x \in R$ is the unique element such that $(x, x_1, \dots, x_n) \in C$. Note that this map is well-defined if $d \geq 2$. Similarly define $\psi' : \widetilde{C^\perp} \to R$ as $(x'_1, \dots, x'_n) \mapsto x'$. We equip $R \oplus R$ with the product $(a, b) \star (c, d) = (ad, bc)$; this defines an R-algebra which we denote A.

Consider the R-submodule of A^n given by

$$D = \{((x_1, x'_1), \dots, (x_n, x'_n)) \mid \mathbf{x} \in \widetilde{C}, \mathbf{x}' \in \widetilde{C^\perp}, \psi(\mathbf{x}) = \psi'(\mathbf{x}')\} \subseteq A^n,$$

and define the map $\psi : D \to R$ by $((x_1, x_1'), \ldots, (x_n, x_n')) \mapsto \psi(\mathbf{x})(= \psi'(\mathbf{x}'))$. We may think of D as the space of consistent sharings, and ψ as the map that reconstructs the secret. For $s \in R$ we write $[s]$ to denote an element of D that maps to s under ψ.

When we use the secret-sharing scheme in the protocol, we also occasionally need to operate on publicly known values. Let $\theta \in \psi^{-1}(1) \subset D$ be a fixed publicly known sharing of $1 \in R$. A public value $x \in R$ can be associated with the canonical sharing $x\theta \in D$.

Now consider the R-module homomorphism $\phi : R^n \to R$ given by $\phi(\mathbf{x}) = -\sum_{i=1}^{n} x_i$. Define the R-submodule of A^n given by

$$M = \{((x_1, x_1'), \ldots, (x_n, x_n')) : \phi(\mathbf{x}) = \phi(\mathbf{x}')\} \subseteq A^n,$$

which intuitively corresponds to redundant additive shares. The reason why we have the redundancy will be made clear in a moment, but at a high level it exists due to the fact that additive shares of the product of two $[\cdot]$-shared secrets can be obtained in two different ways. As we did with D, we define the R-module homomorphism $\phi : M \to R$ given by $((x_1, x_1'), \ldots, (x_n, x_n')) \mapsto \phi(\mathbf{x})(= \phi(\mathbf{x}'))$, and for $s \in R$ we write $\langle s \rangle$ to denote an element of M that maps to s under ϕ.

For $\boldsymbol{x}, \boldsymbol{y} \in A^n$ we define $\boldsymbol{x} * \boldsymbol{y}$ as the point-wise product of these vectors (under the product in A, which is $\star$). We define

$$D^{*2} = \mathsf{span}_R\{\boldsymbol{x} * \boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{y} \in D\} \subseteq A^n,$$

which corresponds at a high level to the operations we performed in the previous paragraphs to obtain additive shares of the product of two secrets.

Proposition 5. *Let $\boldsymbol{x}, \boldsymbol{y} \in D$. Then $\boldsymbol{x} * \boldsymbol{y} \in M$ and moreover $\phi(\boldsymbol{x} * \boldsymbol{y}) = \psi(\boldsymbol{x}) \cdot \psi(\boldsymbol{y})$.*

Proof. Write (x_i, x_i') and (y_i, y_i') for the i-th entry of $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively, for $i = 1, \ldots, n$. The i-th entry of $\boldsymbol{x} * \boldsymbol{y}$ is $(x_i y_i', x_i' y_i)$, via the $\star$-product. There exists $(x_0, x_1, \ldots, x_n) \in C$ and $(y_0', y_1', \ldots, y_n') \in C^\perp$, hence $\sum_{i=1}^{n} x_i y_i' = -x_0 y_0' = -\psi(\mathbf{x})\psi(\mathbf{y}')$. Similarly, there exists $(x_0', x_1', \ldots, x_n') \in C$ and $(y_0, y_1, \ldots, y_n) \in C^\perp$, hence $\sum_{i=1}^{n} x_i' y_i = -x_0' y_0 = -\psi(\mathbf{x}')\psi(\mathbf{y})$. The claim follows. $\square$

In terms of shares, we may write the proposition above as $[x] * [y] = \langle x \cdot y \rangle$. We obtain the following properties.

Theorem 6. *The scheme above $(n - d + 2)$-reconstruction and $(d(\overline{C}^\perp) - 2)$-privacy.*

Proof. ψ is a well-defined R-module homomorphism. Also ψ is surjective, since by Lemma 3 the projection of C onto the zero-th coordinate (corresponding to the secret) is surjective. The map $\phi : D^{*2} \to Z$ is surjective and satisfies $\overline{\psi}(\boldsymbol{x} * \boldsymbol{y}) = \psi(\boldsymbol{x})\psi(\boldsymbol{y})$.

If $U \subseteq \{0, \ldots, n\}$ is an index set of cardinality $d(\overline{C}^\perp) - 2$ then projecting C onto $\{0\} \cup U$ is uniform by Lemma 3, and privacy follows. If $\boldsymbol{x} \in D$ has

$\boldsymbol{x}_U = \mathbf{0}$ for $|U| = n - d + 2$ then since the only codeword in C with weight $\leq n - (n - d + 2) + 1 = d - 1$ is $\mathbf{0}$, we have $\psi(\boldsymbol{x}) = 0$, and reconstruction follows. □

As a corollary, by instantiating these codes with the ones we obtained in Corollary 1, we get our main result.

Theorem 7. *Let $\varepsilon > 0$, and let h be an integer such that $p^h \geq \frac{4}{\varepsilon^2}$. Then there exists a family of R-ASSS $\Sigma_1, \Sigma_2, \ldots$ with $R = \mathsf{GR}(p^k, h)$, such that the number of players $n(\Sigma_i) \to \infty$, and the schemes have $t(\Sigma_i) \geq (1/2 - \varepsilon)n(\Sigma_i)$ privacy and $r(\Sigma_i) \geq (1/2 - \varepsilon)n(\Sigma_i)$ reconstruction.*

We will use the family of arithmetic secret-sharing schemes from the theorem above in the upcoming sections to obtain different multiparty computation protocols.

5 MPC Protocol with Passive Security

In this and the upcoming sections, we fix $\varepsilon > 0$ and consider the Galois ring R of degree $h = \Omega(\log_p(\varepsilon^{-1}))$ over $\mathbb{Z}/p^k\mathbb{Z}$. We consider the family of LSSS over R from Theorem 7. We reuse the notation from Section 4.1: fixing $n \in \mathbb{N}$, we denote by $[x]$ the shares of a secret element $x \in R$, and each of these shares belong to the share space $A = R^2$. We denote by $\langle x \rangle$ shares under the "square" secret-sharing scheme, and recall that given $[x]$ and $[y]$, the parties can perform local computation on their shares to obtain $\langle x \cdot y \rangle$, and we denote this by $\langle x \cdot y \rangle = [x] * [y]$. Whenever we say that parties *reconstruct* a secret $[x]$ (or $\langle x \rangle$), we mean that the parties send their shares to P_1, who uses the reconstruction function to compute x and then sends x to all other parties.

To get a passively secure protocol with perfect security we use the standard approach in MPC of preprocessing some data that can be used to handle multiplication gates efficiently. We follow the template from [16], except that instead of using Reed-Solomon codes, which would lead to a complexity of $O(n \log(n))$, we use our linear secret-sharing scheme $[\cdot]$, allowing us to obtain complexity linear in the number of players.

The techniques from [16] consist, in general, of four main phases:

1. The parties generate "random double-sharings" in a preprocessing phase.
2. The parties use the preprocessed material to distribute inputs.
3. The parties compute the circuit in a gate-by-gate basis. Addition gates are computed locally. Multiplication gates make use of the double-sharings.
4. The output wires are reconstructed towards the parties.

Most of these techniques extend seamlessly to the R setting. The biggest issue lies in the generation of the random double-sharings, which uses a Vandermonde matrix in order to achieve linear complexity, and although these matrices do exist over $R = \mathsf{GR}(p^k, h)$ if $h = \Omega(\log(n))$ [1], our goal here is to avoid this overhead.

In Section 5.1, we show how to get around this issue by moving to a Galois ring extension.

The protocol we describe in the next few subsections proves the following theorem.

Theorem 8. *For every $n, p, k \in \mathbb{N}$, with p a prime, for every $\varepsilon > 0$ and for every arithmetic circuit C over $R = \mathsf{GR}(p^k, h)$ with $h = \Omega(\log_p(\varepsilon^{-1}))$, there exists an n-party MPC protocol that securely computes C against an unbounded semi-honest adversary corrupting up to $t < \left(\frac{1}{2} - \varepsilon\right) \cdot n$ players with a communication complexity of $O(k \cdot \log p \cdot h \cdot |C| \cdot n)$.*

For constant p, k, ε, and by embedding $\mathbb{Z}/p^k\mathbb{Z}$ in R, we obtain the following as a simple corollary.

Theorem 9. *For every $n \in \mathbb{N}$ and for every arithmetic circuit C over $\mathbb{Z}/p^k\mathbb{Z}$ there exists an n-party MPC protocol that securely computes C against an unbounded semi-honest adversary corrupting up to $t < \left(\frac{1}{2} - \varepsilon\right) \cdot n$ players with an amortized communication complexity per multiplication gate of $O(n)$.*

5.1 Offline Phase

As preprocessed material the parties need many shares of the form $([r], \langle r \rangle)$, where $r \in R$ is uniformly random. The basic template used in the literature to achieve this comes from [16], and it uses the fact that Vandermonde matrices are good randomness extractors. However, we cannot use these matrices in our setting since they require the prime p to be at least n, which is not the case for us. Naively, one can use a Galois ring extension in which these matrices exist, as in [1], but this would lose linear complexity. There are two solutions to this problem.

One solution is instead of a hyperinvertible matrix to use the generator matrix of a $[n, u, d]$ linear code over R, with $d \geq t + 1$. This yields u random elements at the cost of n^2 elements of R communicated, which if the rate and distance are linear in n leads to linear complexity. By Theorem 5 we know such codes exist.

The second solution is to move to a Galois ring extension S with high enough Lenstra constant, such that there is a non-singular $n \times n$ Vandermonde matrix. Instead of simply embedding $R \hookrightarrow S$, we use a tensor product $R^s \cong R \otimes_R S \cong S$, where s is the degree of the extension [8,1]. We can take the tensor product of the secret-sharing scheme; the result is a secret-sharing scheme that can be interpreted as s parallel sharings of R. In this way $n - t$ random elements of S can be obtained at the cost of n^2 elements of S communicated. Since each random sharing of S can be interpreted as s random sharings of R, this leads to linear communication per random sharing.

5.2 Online Phase

Now we describe how the parties can securely compute any circuit assuming they have preprocessed enough random sharings $([r], \langle r \rangle)$.

Online Phase

Input Phase. P_i secret-shares its input $x_i \in R$ as follows.

1. The parties take a preprocessed $([r], \langle r \rangle)$ and reconstruct $[r]$ towards P_i.
2. P_i broadcasts the difference $x_i - r$ to all parties.
3. The parties compute $[x_i] = (x_i - r) + [r]$.

Addition Gates. The parties compute locally $[x + y] = [x] + [y]$.

Multiplication Gates. To multiply $[x]$ and $[y]$, the parties use a preprocessed value $([r], \langle r \rangle)$ as follows.

1. The parties compute $\langle x \cdot y \rangle \leftarrow [x] \cdot [y]$.
2. The parties compute $\langle x \cdot y - r \rangle = \langle x \cdot y \rangle - \langle r \rangle$ and reconstruct this value.
3. The parties compute $[x \cdot y] = [r] + (x \cdot y - r)$.

Output Wires. For every shared output wire $[w]$, the parties reconstruct w.

The complexity of the protocol above is dominated by the reconstructions in the multiplication gates. Each such reconstruction involves sending $O(n)$ elements in A. Since these elements have bit-length $O(k \cdot \log(p) \cdot h)$, the overall complexity of these reconstructions is $O(k \cdot \log(p) \cdot h \cdot |C| \cdot n)$.

5.3 Active Security with Abort

Even though we present an actively secure protocol with guaranteed output delivery in Section 6, it is still worth mentioning that a much simpler protocol can be envisioned if one is aiming for security with abort.

Our starting observation is that the online multiplication protocol presented previously is secure up to additive attacks, as defined in [19], or, put more precisely, the only attack that an active adversary can carry out is to cause the result of the multiplication to be wrong by an additive amount that is known by him and that is completely independent of the inputs. To see why this is the case, we observe that if the preprocessed pair $([r], \langle r \rangle)$ is correctly shared, then the only thing that the adversary can do in the online phase is broadcasting[12] an incorrect difference $r - xy + \delta$ (assuming that P_1 is corrupted), but the effect of this is that the final shares the parties get are $[xy + \delta]$, which constitutes an additive attack. Furthermore, the preprocessed pairs can be guaranteed to be consistent by a simple extension to the preprocessing protocol in Section 5.1 that adds a consistency check at the end (for instance as in done in [1] or in [11]).

Very recently it was shown in [2] how to compile *any* protocol over rings that is secure up to additive attacks to an actively secure protocol. Given that

[12] To handle the active case we must have a proper broadcast channel, that is, we need to assume a the existence of a broadcast functionality. This is required in the setting of honest majority setting with statistical security, that is, a statistically secure protocol that instantiates a broadcast functionality cannot exist [22].

our multiplication protocol satisfies this condition (and it can be verified that it satisfies the other conditions required by the compiler), we obtain an actively secure protocol by feeding our protocol from the previous section through the compiler from [2]. The resulting protocol has linear communication in the number of parties.

6 Active Security with Guaranteed Output Delivery

The main theorem we prove in this section is the following.

Theorem 10. *For every $n, p, k \in \mathbb{N}$, with p a prime, for every $\varepsilon > 0$ and for every arithmetic circuit C over $R = \mathsf{GR}(p^k, h)$ with $h = \Omega(\log_p(\varepsilon^{-1}))$, there exists an n-party MPC protocol that securely computes C with guaranteed output delivery against an unbounded active adversary corrupting up to $t < \left(\frac{1}{2} - \varepsilon\right) \cdot n$ players, with negligible failure probability in $\kappa \in \mathbb{N}$, offline communication complexity of $O(k \cdot \log p \cdot (h \cdot |C| \cdot n \cdot \log(n) + n^7 \cdot \kappa))$, and online communication complexity of $O(k \cdot \log p \cdot h \cdot |C| \cdot n)$.*

Typically, we regard p, k and ε (and therefore h) as constants, so that the only variables are n, C and κ. In this case, we see that the amortized complexity per multiplication is $O(n)$ for the online phase, and $O(n \log(n))$ for the offline phase. Furthermore, computation over $\mathbb{Z}/p^k\mathbb{Z}$ can be obtained by embedding the computation into a Galois ring R of constant degree h, and adding a check of input correctness as in [1]. The following theorem is thus obtained as a corollary.

Theorem 11. *For every constants $p, k \in \mathbb{N}$, with p a prime, every constant $\varepsilon > 0$, and for every arithmetic circuit C over $\mathbb{Z}/p^k\mathbb{Z}$ there exists an n-party MPC protocol that securely computes C with guaranteed output delivery against an unbounded active adversary corrupting up to $t < \left(\frac{1}{2} - \varepsilon\right) \cdot n$ players, with negligible failure probability in κ, amortized offline communication complexity of $O(n \log(n))$ per multiplication gate and amortized online communication complexity of $O(n)$ per multiplication gate.*

The rest of this section is devoted to proving Theorem 10. We do so by adapting the protocol from [5] over fields, which we refer to as the BFO protocol, to work over a Galois ring R, while also making use of our LSSS from Section 4. Due to space constraints, we only detail the most essential modifications to the BFO protocol, and assume some of the terminology from [5] as given. An overview of the BFO protocol and more details are provided in Section 6.1 below.

In order to extend the BFO protocol to our setting while preserving its efficiency, we mostly need to adapt the preprocessing phase. Arguments regarding dispute control carry over immediately, since they are essentially combinatorial in nature. In the next sections we discuss how to adapt the preprocessing: the verification of multiplication triples is in Section 6.6, and the computation of the tags is sketched in Section 6.5. Additionally, the fact that these tags provide the required authentication features when instantiated over Galois rings is not trivial, and we discuss this thoroughly in Section 6.5.

We stress that our goal here is not to present a full-fledged self-contained MPC protocol, but rather to describe our novel techniques and extensions to the BFO protocol. Hence, although we present a brief overview of the BFO protocol below, we assume some familiarity with the work of [5] and we omit most of its heavy machinery, especially everything that extends seamlessly to Galois rings. We also remark that, even though we assume the existence of a broadcast channel implicitly (as the dispute control layer requires it), our complexity analysis does not include the cost of these broadcasts, which is equal to the corresponding cost in [5] and is independent of the circuit size.

Finally, we notice that the techniques from [21], which improve the complexity of the protocol from [5] by removing an additive term of $n^2 d$, where d is the depth of the circuit, rely mostly on the batch triple check from [5], which we extend in Section 6.6 to the Galois ring setting. Hence, the optimizations from [21] can be also applied to Galois rings, resulting in a much more efficient protocol that does not have a quadratic communication complexity in terms of the numbers of parties and the depth of the circuit.

6.1 Overview of the BFO Protocol

We base our work in Section 6 on the protocol from [5], henceforth referred to as the BFO protocol, which is set in the field setting with honest majority ($t < n/2$), and has near-linear amortized communication complexity, or more precisely, the complexity is linear but it has an overhead of $\log(n)$ for both the offline and online phases due to the use of Shamir secret sharing.

In this section we provide a quick overview of the BFO protocol. It uses Shamir secret sharing and it follows an offline-online paradigm, similar to the protocol from Section 5, in which multiplication triples are preprocessed to be used later on to compute the multiplication gates of the circuit. However, the main complication arises from the fact that errors in the shares can be detected but not *corrected*, which is an issue for obtaining guaranteed output delivery. To this end, the BFO protocol uses the *dispute control* techniques from [3], which allow the parties to generate disputes whenever an inconsistency is found so that the current computation can be repeated whilst guaranteeing that the same dispute will not occur again. Furthermore, another complication appears due to the complexity goal of the BFO protocol.

The overview of the BFO protocol we provide next is largely based on [5, Section 2.6], and we refer the reader to that reference (and in general to the complete [5] work) in order to acquire familiarity with the BFO protocol. Here we do not aim to be self-contained nor explicit about this protocol. On the one hand, the protocol is already complex enough as to describe it in detail here. Secondly, to keep the focus on our novel techniques we concentrate only on the aspects from the BFO protocol that need to be adapted/modified to our Galois ring setting.

Dispute Control. Every party maintains a list of parties that it is in dispute with. The computation proceeds in *segments*, and the adversary may cause any given segment to fail. However, whenever this happens a *fault localization* procedure is executed so that at the end a new dispute is generated, where at least one of the two parties in the dispute is corrupt. At this point the segment can be re-executed, and, furthermore, it is guaranteed that the same dispute cannot happen. As a result, the adversary may cause a segment to be repeated at most n^2 times. By dividing whole computation in n^2 segments, the adversary may only cause an overhead of 2 to the execution.

Preparation Phase. The preparation phase is in charge of generating certain shared material that will be used later on in the online phase. This consists of the following:

– *Two-level shared multiplication triples.* These are triples $\{[\![a]\!]_R, [\![b]\!]_R, [\![c]\!]_R\}$ where $c = a \cdot b$. The two-level sharings $[\![\cdot]\!]$ are defined in Section 6.2.
– *Local base sharings.* Each party P_i has a set of local sharings, that consist of $[\cdot]$-shares where P_i knows the underlying secret. These shares are used to obtain the two-level shares $[\![\cdot]\!]$, which are the ones that are actually used for the computation. At a high level, two-level shares satisfy that each share is a local base share, so there is enough "redundancy" in case that an opening fails (i.e. having shares of the shares allows the party to "detect" who send a wrong share at reconstruction time).
– *Authentication tags.* Two-level sharings are not enough, however. To avoid parties from lies about their one-level shares $[\cdot]$, authentication tags are put in place. At a high level, each share in $[\cdot]$ is authenticated towards each party with a one-time-key MAC scheme, so that lying about these shares is detected by honest parties.

A total of $L = O(|C|)/n$ local base sharings per party are required. To compute these, each party distributes shares that are then checked for consistency. This is also known as a verifiable secret sharing (VSS) protocol. We show how to extend this to our ring setting in Section 6.4.

A total of $M = O(|C|)$ multiplication triples must be processed. These are computed passively (e.g. using a protocol that resembles closely our protocol from Section 5), and then their correctness is checked using a novel check that saves in communication. Following the dispute control technique, these triples are computed and checked across n^2 segments, each one containing $m = M/n^2$ multiplication triples. In Section 6.6 we show how to perform this check over our Galois ring R.

Finally, the computation of the tags is done by exploiting certain symmetric properties of Shamir secret sharing, namely that if $(x_1, \dots, x_n)$ are Shamir-shares of a secret x, then $(x, x_1, \dots, x_{i-1}, x_{i+1}, x_n)$ are Shamir-shares of x_i, but with different evaluation points. We show in Section 6.2 that this property is more general, and that it translates to our LSSS over Galois rings, and hence we can use the techniques from BFO for the computation of the tags as sketched in Section 6.5.

Input Phase. The distribution of the inputs happens in a very similar way as in our protocol from Section 5.3: A random share $[\![r]\!]$ is computed (it can be taken from a preprocessed multiplication triple) and opened towards the input provider P_i, who, on input x, broadcasts $x - r$. The parties finally compute $[\![x]\!] = [\![r]\!] + (x - r)$.

Computation Phase. The online phase makes use of the triples to process multiplication gates. This requires the parties to reconstruct certain shares, and this can be disrupted by the adversary if it sends incorrect shares. It turns out that these shares that must be reconstructed can be seen as (public) linear combinations of the local base sharings, which are authenticated, so players who lie about their shares can be caught.

6.2 Different Types of Shares

From now on, we fix $R = \mathsf{GR}(p^k, h)$ and $S = \mathsf{GR}(p^k, \kappa)$. Notice that we may view S as an extension of R of degree κ/h. The BFO protocol follows the template from Section 5, except that it has an additional mechanism to ensure that whenever the adversary cheats this can be detected and the computation can continue. This is achieved by using different types of secret-sharings (especially 2-level sharings, defined below), which create enough "redundancy" for the parties to be able to interactively[13] correct any error the adversary may introduce.

The multiple types of sharings considered for our extension of the BFO protocol are found below — for the intuition on these definitions we refer the reader to [5]. Note that these sharings were originally defined purely in the context of Shamir's secret-sharing scheme. We plug in the family of LSSS over R from Theorem 7 and get a more general setting: not only because our LSSS is defined over a Galois ring, but also because it does not have information rate 1, i.e., the shares do not have the same size as the secret.

Single sharing. These are the sharings $[x]$ as defined using our LSSS. The secret space is R, and the share space is A. They are the analogue to the degree-t Shamir sharings from [5].
Square sharing. These are the shares $\langle x \rangle$ under the "square" secret-sharing scheme. The secret space is R, and the share space is A. As in Section 5, they are the analogue to the degree-$2t$ Shamir sharings from [5].
Twisted single sharing. These are defined with respect to a coordinate $i \in \{1, \ldots, n\}$. Let $x \in A$. We denote by $\lceil x \rfloor^i$ an element $\boldsymbol{x} = (x_1, \ldots, x_n) \in D$ such that $\psi(\boldsymbol{x}) = 0$ and $x_i = x$. One may view this as a sharing of 0 such that i-th share equals x.
Twisted square sharing. These are defined with respect to a coordinate $i \in \{1, \ldots, n\}$. Let $x \in A$. We denote by $\langle x \rangle^i$ an element $\boldsymbol{x} = (x_1, \ldots, x_n) \in M$ such that $\phi(\boldsymbol{x}) = 0$ and $x_i = x$. One may view this as square sharing of 0 such that the i-th share equals x.

[13] In contrast to the $t < n/3$ case in which an appropriate choice of the code allows for non-interactive error correction.

Two-level single sharing. The secret space is R and the share space is A^n. For $x \in R$, we define $[\![x]\!]$ as an $n \times n$ matrix $(x_{i,j})_{i=1,\dots,n}^{j=1,\dots,n} \in A^{n\times n}$, such that:

1. The j-th share is the j-th column.
2. Each i-th row $\boldsymbol{x}_i = (x_{i,1}, \dots, x_{i,n})$ is a vector in D, i.e., it constitutes a single sharing $[x_i]$ of some element $x_i \in R$.
3. We have $x_1 + \cdots + x_n = x$.

Two-level square sharing. Denoted $\langle\!\langle x \rangle\!\rangle$. It is identical to a two-level single sharing, except the rows are vectors in M, and hence constitute square sharings $\langle x_i \rangle$.

6.3 Secret Sharing over a Galois Ring Extension

In [5] a large field is needed for some subprotocols in order to ensure that the probability of cheating is low. To this end, an appropriate field extension is used but in such a way that the overall complexity is not affected. In the BFO protocol where Shamir secret sharing is used, it is straightforward to use shares defined over the base field and the extension field together, since the arithmetic is compatible. In our case, we use a Galois ring extension and we show that the arithmetic is indeed compatible as well.

Let L be a Galois ring extension of R of degree r. Intuitively, the secret sharing scheme $[\cdot]$ (and similarly for $\langle\cdot\rangle$) over R can be extended to L as follows: The shares of an element $\alpha \in L$ denoted by $[\alpha]_L$, are obtained by writing α as $\alpha = \sum_{i=1}^{r} a_i \cdot \omega_i$ where $a_i \in R$ and $\{\omega_i\} \subseteq L$ is a basis of L over R and then setting $[\alpha]_L := ([a_1], \dots, [a_r])$. This secret sharing scheme over L has share space A^r, and it inherits the privacy and reconstruction properties of $[\cdot]$. To formalize this intuition, we have to rely on the theory of commutative algebra.

We begin with some basic lemmas.

Lemma 4. *Let R be a ring and let $L \supseteq R$ be an R-algebra. If M is an R-module, then extension of scalars of M over L, defined as*

$$\mathsf{span}_L(M) := L \otimes_R M,$$

is an L-module under the product $x \cdot (y \otimes m) = (xy) \otimes m$. We may think of M as a subset of $\mathsf{span}_L(M)$ via $m \mapsto 1 \otimes m$.

For our case in which L is a Galois ring extension of R of degree r and $M \subseteq R^\ell$, we can visualize the L-module $\mathsf{span}_L(M)$ as the L-submodule of L^ℓ obtained by taking all possible linear combinations of elements in M with scalars in L (notice that the product between a scalar in L and a vector in M is well defined since $M \subseteq R^\ell \subseteq L^\ell$).

The following lemma shows that linear maps can be extended under the span operator.

Lemma 5. *Let R be a ring and let M and N be R-modules. Consider an R-module homomorphism $f : M \to N$. If $M_L = \mathsf{span}_R(M)$ and $N_L = \mathsf{span}_R(N)$, then the map $f_L : M_L \to N_L$ given by $x \otimes m \to x \otimes f(m)$ (and extending by linearity) is an L-module homomorphism and $f_L\,|_M = f$.*

With these tools at hand, we can proceed to the formal extension of our LSSS over R to L. We begin by extending $[\cdot]$, but a similar analysis can be done for $\langle\cdot\rangle$. Recall that the shares $[\cdot]$ are elements of the R-submodule of A^n

$$D = \{((x_1, x_1'), \dots, (x_n, x_n')) : \mathbf{x} \in \widetilde{C}, \mathbf{x}' \in \widetilde{C^\perp}, \psi(\mathbf{x}) = \psi'(\mathbf{x}')\} \subseteq A^n,$$

where $A = R^2$ and $C \subseteq R^{n+1}$ is a $[n+1, t, d]$-linear code $C \subseteq R^{n+1}$ over R with dual distance $d^\perp$, and $\widetilde{C} \subseteq R^n$ is the projection of C onto its last n coordinates, and similarly for $\widetilde{C^\perp} \subseteq R^n$. Furtermore, recall that there is a reconstruction R-module homomorphism $\psi : D \to R$.

Consider now the L-modules $D_L = \mathsf{span}_L(D)$ and $A_L = \mathsf{span}_L(A)$. Using lemmas 4 and 5, we may consider the L-module homomorphism $\psi_L : D_L \to L$ that extends ψ from D to D_L. Given $\alpha \in L$, we write $[\alpha]_L$ to denote an element of D_L that maps to α under ψ_L. Again, for concreteness, we may think of L as R^r and A_L as A^r, and sharing an element $\alpha \in L$ amounts to sharing each of the r coordinates using $[\cdot]$. Furthermore, since $D \subseteq D_L$, it holds for $x \in R$ that shares $[x]$ can be automatically seen as $[x]_L$.

A similar analysis can be done for the LSSS $\langle\cdot\rangle$. The resulting LSSS over L is denoted by $\langle\cdot\rangle_L$, and its reconstruction function is denoted by $\phi_L : M_L \to L$, where $M_L = \mathsf{span}_L(M)$ (reusing the notation from Section 4.1).

Finally, we need to show that the multiplicative property also holds for these new types of shares over L. This is done in the following proposition, which can be seen as an analogue version of Proposition 5

Proposition 6. *Let $\boldsymbol{\alpha}, \boldsymbol{\beta} \in D_L$. Then $\boldsymbol{\alpha} * \boldsymbol{\beta} \in M_L$ and moreover $\phi_L(\boldsymbol{\alpha} * \boldsymbol{\beta}) = \psi_L(\boldsymbol{\alpha}) \cdot \psi_L(\boldsymbol{\beta})$.*

Proof. This proposition can be proved by means of the universal property of tensor products, but we choose are more direct approach to avoid introducing such tool. Write $\boldsymbol{\alpha} = \sum_{i=1}^{n_1} u_i \otimes \boldsymbol{x_i}$ and $\boldsymbol{\beta} = \sum_{j=1}^{n_2} v_j \otimes \boldsymbol{y_j}$ where $u_i, v_j \in L$ and

$\boldsymbol{x_i}, \boldsymbol{y_j} \in D$. We have that

$$\begin{aligned}
\phi_L(\boldsymbol{\alpha} * \boldsymbol{\beta}) &= \phi_L\left(\left(\sum_i u_i \otimes \boldsymbol{x}_i\right) * \left(\sum_j v_j \otimes \boldsymbol{y}_j\right)\right) \\
&= \phi_L\left(\sum_{i,j} u_i v_j \otimes \boldsymbol{x}_i * \boldsymbol{y}_j\right) \\
&= \sum_{i,j} \phi_L\left(u_i v_j \otimes \boldsymbol{x}_i * \boldsymbol{y}_j\right) \\
&= \sum_{i,j} u_i v_j \otimes \phi\left(\boldsymbol{x}_i * \boldsymbol{y}_j\right) \\
&= \sum_{i,j} u_i v_j \otimes \psi(\boldsymbol{x}_i) \cdot \psi(\boldsymbol{y}_i) && \text{(By Proposition 5)} \\
&= \sum_{i,j} (u_i \otimes \psi(\boldsymbol{x}_i)) \cdot (v_j \otimes \psi(\boldsymbol{y}_i)) \\
&= \left(\sum_i u_i \otimes \psi(\boldsymbol{x}_i)\right) \cdot \left(\sum_j v_j \otimes \psi(\boldsymbol{y}_j)\right) \\
&= \left(\sum_i \psi_L(u_i \otimes \boldsymbol{x}_i)\right) \cdot \left(\sum_j \psi_L(v_j \otimes \boldsymbol{y}_j)\right) \\
&= \psi_L\left(\sum_i u_i \otimes \boldsymbol{x}_i\right) \cdot \psi_L\left(\sum_j v_j \otimes \boldsymbol{y}_j\right) = \psi_L(\boldsymbol{\alpha}) \cdot \psi_L(\boldsymbol{\beta}).
\end{aligned}$$

With these tools at hand, it is easy to prove that our construction constitutes an Arithmetic LSSS, according to Definition 3. Furthermore, we notice that the different types of sharings defined in Section 6.2 extend naturally the the Galois ring extension setting.

Remark 2. Since L is itself a Galois ring, one may consider using the theory from Section 4.1 to obtain an Arithmetic LSSS over L. This naturally works, but in our setting we want to consider two Galois rings $R \subseteq L$, and we want the two LSSS over these rings to be "compatible". As an analogy, consider Shamir LSSS over a field $\mathbb{F}_q$. This LSSS is not necessarily compatible with Shamir LSSS over $\mathbb{F}_{q^r}$ if the evaluation points are picked in $\mathbb{F}_{q^r}$. However, if the points are in $\mathbb{F}_q$, which corresponds to performing our LSSS extension as above, the two schemes are compatible, in the sense that shares over $\mathbb{F}_q$ can be seen as shares over $\mathbb{F}_{q^r}$.

6.4 Verifiable Secret Sharing

Our first task is to guarantee that whenever corrupt dealer distributes shares, these are consistent, that is, that they belong to the R-module D. More specifically, VSS

Output: ℓ consistent $[\cdot]$-sharings.

Dealing. The dealer P_d executes the following.
1. Sample $\sigma_1, \ldots, \sigma_h \in S$ and $\chi \in S$ uniformly at random, with $h = \frac{\ell \cdot d}{\kappa}$.
2. Distribute $[\chi]_S, [\sigma_1]_S, \ldots, [\sigma_h]_S$ to the parties.

Verification. The parties execute the following in order to verify that the shares are consistent.
1. Use coin-tossing to sample a random element $\lambda \in S$.
2. The parties compute $[x]_S = [\chi]_S + \sum_{i=1}^{h} \lambda^i \cdot [\sigma_i]_S$ and send their shares to every other player.
3. If these shares are inconsistent, then the parties execute a dispute-control-based fault localization procedure. Otherwise they parse $[\sigma_1]_S, \ldots, [\sigma_h]_S$ as $h \cdot \frac{\kappa}{d} = \ell$ shares over R (see Section 6.3) and output these shares as consistent.

Fig. 1. Verifiable Secret Sharing

suppose that some dealer has distributed some shares, the task consists then on checking that these are consistent. This is needed for obtaining the $L = O(|C|)/n$ required local base sharings, and following the dispute control paradigm, this is divided into n^2 segments of length $\ell = L/n^2$ each. For reference, this corresponds to protocol VerShare in [5].

We present our protocol for checking correctness of the shares in Fig. 1. We remark that the resulting protocol provides the same functionality as VerShare in [5], except that we do not consider explicitly the dispute control layer needed in order to ensure guaranteed output delivery.

At a high level, the protocol follows the same template as many protocols in the literature for the same task (e.g. [3,16,5]) of taking a masked linear combination of the shares and then checking the correctness of the resulting sharings. This works over fields, but the argument typically uses the fact that non-zero elements over a field are invertible, and it is not clear how to generalize such arguments to our Galois rings. Fortunately, in [1] the authors showed how to extend such check in the case of Shamir secret sharing over Galois rings. This is achieved due to the following lemma, which shows that Galois rings, albeit not being fields, share many common features with them.

Lemma 6 (Lemma 2 in [1]). *Let $f \in R[X]$ polynomial of arbitrary degree $\ell > 0$. Then $\Pr_{x \leftarrow R}[f(x) = 0] \leq \frac{\ell}{p^\kappa}$, where x is drawn uniformly from R.*

Furthermore, the VSS protocol in [1] introduces an optimization that lowers the cheating probability by "packing" several Galois ring elements over R into one single element of S. Our protocol exploits this feature as well, but we need to handle first the technical issue that their VSS protocol is set in the Shamir secret sharing setting, whereas here we have a more general and abstract LSSS $[\cdot]$.

We solve this issue via the following lemma about free submodules of R^k, which shows that even if modules over arbitrary commutative rings are difficult to handle, they are not quite so over a Galois ring. More specifically, it shows that bases for submodules of modules over Galois rings can be extended to a basis of the module, which is the abstract property that guarantees that guarantees the security of the VSS protocol in [1]. This property holds mostly due to the fact that Galois rings are local and therefore they share many characteristics with fields.

Lemma 7. *Let $R = \mathsf{GR}(p^k, r)$ be a Galois ring. Given a free R-module $M \subseteq R^n$ of rank ℓ, there exists a free R-module $N \subseteq R^n$ such that $R^n = M \oplus N$.*

Proof. Let $v_1, \dots, v_\ell \in R$ be a basis of M, and let us denote $\overline{v}_i = v_i \bmod p$. Consider $\overline{M} \subseteq \mathbb{F}^n$, the vector space over $\mathbb{F}$ generated by $\{\overline{v}_i\}_{i=1}^{\ell}$. By classical theorems in linear algebra, we know there exist $\{v_j\}_{j=\ell+1}^{n} \subseteq \mathbb{F}^n$ such that $\{\overline{v}_i\}_{i=1}^{\ell} \cup \{v_j\}_{j=\ell+1}^{n}$ is a basis of $\mathbb{F}^n$.

Now we claim that $\{v_i\}_{i=1}^{n}$ is a basis for R^n, which, by setting N to be the R-module generated by $\{v_j\}_{j=\ell+1}^{n}$, concludes the proof. It is clear that these elements generate R^n since their reduction generates the residue field $\mathbb{F}$, hence, it suffices to show that they are R-linearly independent. To this end, suppose that $0 = \sum_{i=1}^{n} \lambda_i \cdot v_i$ for some $\lambda_i \in R$. By taking modulo p and using the fact that $\{\overline{v}_i\}_{i=1}^{n}$ is a basis of $\mathbb{F}^n$, we have that $\lambda_i \equiv 0 \bmod p$ for $i = 1, \dots, n$. We can write then $0 = p \cdot (\sum_{i=1}^{n} \lambda'_i \cdot v_i)$, but in this case it must be the case that $\sum_{i=1}^{n} \lambda'_i \cdot v_i = 0 \bmod p^{k-1}$, so the process can be iterated to show that $\lambda_i = 0$ for all i, which concludes the proof. □

With this lemma we can prove the following proposition, which shows that the space of all possible shares can be divided as a direct sum of consistent and inconsistent shares.

Proposition 7. *There exists an S-submodule $K \subseteq (A_S)^n$ such that $(A_S)^n = D_S \oplus K$*

Proof. First we observe that it suffices to show this for $S = R$, that is, it suffices to show that there exists an R-submodule $N \subseteq A^n$ such that $A^n = D \oplus N$. The claimed result would follow then by setting $K = \mathsf{span}_S(N)$.

Now, to show the existence of N, we recall that $A^n \cong R^{2n}$ as R-modules, and simply apply Lemma 7 with $M = D$.[14] □

Now we proceed to the security proof of the VSS protocol. We notice that privacy holds trivially due to the fact that $\chi \in R$ is uniformly random, so it masks the linear combination of the secrets. Hence we only focus on the soundness of the protocol.

Theorem 12. *If the check at the end of the VSS protocol in Fig. 1 passes, then the outputted shares are consistent with probability at least $1 - \frac{\ell \cdot d}{\kappa \cdot p^\kappa}$.*

[14] The fact that D is free is not trivial but it is not hard to prove either. We omit this detail for simplicity.

Proof. We begin by noticing that the final shares over R are consistent if and only if the shares $[\sigma_1]_S, \dots, [\sigma_h]_S$ are consistent, so we focus on showing this instead. By definition, $[\sigma_i]_S$ is consistent if and only if $[\sigma_i]_S \in D_S$, where $D_S = \mathsf{span}_S(D)$. Furthermore, since by assumption the check at the end of the protocol passed, it holds that $[\chi]_S \in D_S$.

Now, consider an S-submodule $K \subseteq A^{n+1}$ such that $A^{n+1} = D_S \oplus K$, which exists thanks to Proposition 7, and let us write each $[\sigma_i]_S$ as $[\sigma_i]_S = \boldsymbol{\alpha}_i + \boldsymbol{\beta}_i$, and $[\chi]_S = \boldsymbol{\alpha}_0 + \boldsymbol{\beta}_0$, where $\boldsymbol{\alpha}_j \in D_S$ and $\boldsymbol{\beta}_j \in K$ for $j \in \{0, \dots, h\}$. Our goal is to show that each $\boldsymbol{\beta}_i$ is zero, since this implies that $[\sigma_i]_S \in D_S$ for all i.

Notice that

$$\underbrace{[\chi]_S}_{\in D_S} = \underbrace{\sum_{i=0}^{h} \lambda^i \cdot \boldsymbol{\alpha}_i}_{\in D_S} + \underbrace{\sum_{i=0}^{h} \lambda^i \cdot \boldsymbol{\beta}_i}_{\in K},$$

which implies that $\sum_{i=0}^{\ell} \lambda^i \cdot \boldsymbol{\beta}_i = 0$.

Finally, consider the polynomial in $A^n[X]$ given by $\sum_{i=0}^{h} X^i \cdot \boldsymbol{\beta}_i$, which corresponds to one polynomial in $R^{2n}[X]$. If at least one of these $2n$ polynomials is not the null polynomial, we can apply Lemma 10 to obtain that $\sum_{i=0}^{h} \lambda^i \cdot \boldsymbol{\beta}_i = 0$ can only hold with probability at most h/p^κ. Hence, we conclude that with probability at least $1 - h/p^\kappa$ these polynomials are zero, which implies that $\boldsymbol{\beta}_i = 0$ for $i = 1, \dots, h$, concluding the proof. $\square$

Complexity analysis. The complexity of the VSS protocol for one segment without the fault localization layer is $O(k \cdot \log_2(p) \cdot (n \cdot \ell \cdot d + n^2 \cdot \kappa))$. This is essentially the same complexity as in [5], but without the $\log(n)$ overhead. In the actual protocol this is called once per party, which adds a factor of n, and since $\ell = O(|C|)/n^3$, this complexity translates into $O(k \cdot \log_2(p) \cdot (|C| \cdot n^{-1} \cdot d + n^2 \cdot \kappa))$. Since this for one single segment, multiplying by the n^2 segments we get $O(k \cdot \log_2(p) \cdot (|C| \cdot n \cdot d + n^4 \cdot \kappa))$.

6.5 Authentication Tags

In the BFO protocol, whenever some cheating is detected, parties resort to dispute control in order to partially identify the cheater. One of the critical points in which the adversary can cheat in the protocol is when sending shares in order to reconstruct shared values, since in principle any corrupt party can lie about its own share. In order to be able to detect who sent a wrong share, the parties need an additional mechanism that somehow "binds" a party to its own share. This is precisely the purpose of the two-level shares defined in Section 6.2: the share of each party P_i is also shared among the other parties, so the parties can check whether P_i is lying about its share by reconstructing it from the two-level shares.

Unfortunately, nothing prevents the parties to also lie in the reconstruction of the two-level shares themselves. In order to deal with this situation, authentication tags are put in place, which allow a party to announce a share and prove that it is *correct*, or more precisely, prove that it is the same share that was created at the beginning of the protocol, which was guaranteed to be correct.

At a high level, the tags over fields in the BFO protocol work as follows.[15] Consider a value $s \in \mathbb{F}_q$ that is shared as $[s] = (s_1, \ldots, s_n) \in \mathbb{F}_q^n$ using Shamir LSSS. Player P_i holds share $s_i \in \mathbb{F}_q^n$, and to prevent him from lying about his share, P_i is given a *tag* $\tau = \mu \cdot s_i + \nu$, where the key $\mu, \nu \in \mathbb{F}_q^n$ is random and only known by some verifier P_j. At the time of opening, P_i has to present a share $s_i' = s_i + \delta$ plus a tag $\tau' = \tau + \Delta$, where $\delta, \Delta \in \mathbb{F}_q$ may be nonzero for the case of a corrupt P_i, and the verifier P_j checks whether $\tau' \overset{?}{=} \mu \cdot s_i' + \nu$. This check passes if and only if $\Delta = \mu \cdot \delta$. If P_i attempts to cheat (i.e., $\delta \neq 0$) and if the verifier P_j is honest, then P_i does not know the random μ, and therefore check must fail with high probability (assuming the field is large). This can be seen by using that $\delta \neq 0$ is invertible, so $\Delta \cdot \delta^{-1} = \mu$, which due to the randomness of μ cannot be satisfied.

Adapting this to our setting is not straightforward because of two reasons. First, Galois rings are not fields for $k > 1$ and therefore the argument above does not apply directly, since $\delta \neq 0$ need not be invertible. Fortunately, using the ideas from [1] we still can show that the equation $\Delta = \mu \cdot \delta$ holds with negligible probability. However, the second issue is more delicate and it has to do with the fact that in our setting each share in $[s]$ is not a single Galois ring element but it is actually an element of $A = R^2$.

We handle this second issue by extending the authentication scheme from above not only from $\mathbb{F}_q$ to R, but to A. At a high level, the tag corresponding to a share $s_i \in A$ is computed as $\tau = \mu \star s_i + \nu \in A$, for the key $\mu, \nu \in A$. Cheating in this new MAC scheme corresponds to solving equations of the form $\Delta = \mu \star \delta$, for some $\Delta, \delta \in A$, which intuitively cannot be satisfied since it corresponds to two similar equations over R. We develop the details in what follows.

Definition and properties of the tags. We use the same template as the MAC scheme from [5], which authenticates batches instead of individual values. Let $\{(s_{j,1}, \ldots, s_{j,\kappa/h})\}_{j=1}^{\ell} \in (R^h)^\ell$. Recall from Proposition 3 that $R^{\kappa/h} \cong S$, so we may think of each $(s_{j,1}, \ldots, s_{j,\kappa/h})$ as one single element $\sigma_j \in S$. Following Section 6.3, we consider shares $[\sigma_j]_S$ which can be obtained by sharing each of its coordinates as $[s_{j,i}]$. By writing $[s_{j,i}] = (s_{j,i,1}, \ldots, s_{j,i,n}) \in D^n$ and considering the vector $(s_{j,1,w}, \ldots, s_{j,\kappa/h,w}) \in A^{\kappa/h}$ for $j \in \{1, \ldots, \ell\}$ and $w \in \{1, \ldots, n\}$, which we identify with an element $\sigma_{j,w} \in A_S$ where $A_S = \mathsf{span}_S(A)$, we can see that $[\sigma_j]_S = (\sigma_{j,1}, \ldots, \sigma_{j,n}) \in (A_S)^n$.

Notice that the S-algebra A_S can be seen simply as S^2, with the product operation defined as $(\alpha, \alpha') \star (\beta, \beta') = (\alpha \cdot \beta', \alpha' \cdot \beta)$. With this in hand we can define what it means for the shares of σ to be authenticated.

[15] As we will see, the scheme is a bit more complex since the values are tagged in blocks rather than individually, but we will not consider this for now.

Definition 4. *(Informal.)*[16] *We say that the* $\ell \cdot \frac{\kappa}{h}$ *shares* $\{[s_{j,i}]\}_{j=1,i=1}^{\ell,\kappa/h}$ *are* authenticated *if for every pair of players* P_u, P_v *the following holds:*

- P_v *has a random key* $\boldsymbol{\mu} \in (A_S)^\ell$ *and* $\nu \in A_S$*.*
- P_u *has a tag* $\tau \in A_S$
- $\tau = \boldsymbol{\mu} \odot \boldsymbol{\sigma} + \nu$*, where* $\boldsymbol{\sigma} = (\sigma_{1,u}, \ldots, \sigma_{\ell,u}) \in (A_S)^\ell$ *and* $\odot$ *denotes the dot product operator.*

Proposition 8 argues that the tags defined above serve their purpose, i.e. a corrupt P_u cannot lie about any of his shares $s_{j,i,u}$ and still present a valid tag without an honest P_v detecting this. The proof follows a similar argument as the one sketched before over fields for the BFO protocol. However, we first need to show that the S-algebra A_S, even though it is not a field, and not even a Galois ring, does have good properties in terms of roots of linear equations. This is shown in the following lemma, which can be seen as an analogue of Lemma 10 to the S-algebra A_S, but considers multivariate polynomials of degree 1.

Lemma 8. *Let* $L = \mathsf{GR}(p^k, r)$ *and let* $B = L^2$ *be the* L*-algebra with multiplication given by* $(\alpha, \alpha') \star (\beta, \beta') = (\alpha\beta', \alpha'\beta)$*. Let* $\boldsymbol{\alpha} \in B^\ell$ *and* $\gamma \in B$*. If* $\boldsymbol{\alpha} \neq 0$*, then* $\Pr_{\boldsymbol{\beta} \leftarrow B^\ell}[\boldsymbol{\alpha} \odot \boldsymbol{\beta} = \gamma] \leq \frac{\ell}{p^r}$*.*

Proof. Suppose that $(\alpha_1, \ldots, \alpha_\ell) \odot (\beta_1, \ldots, \beta_\ell) = \gamma$, and suppose that $\boldsymbol{\alpha} \neq 0$. Without loss of generality, assume that $\alpha_1 \neq 0$, so $\alpha_1 \star \beta_1 = \rho$, with $\rho = \gamma - \sum_{j=2}^{\ell} \alpha_j \star \beta_j$. Let π_1, π_2 be the canonical L-algebra homomorphisms $B \to L$ of projection onto the first and second coordinate, respectively. Since $\alpha_1 \neq 0$, for at least one of $i = 1$ or $i = 2$ we have $\pi_i(\alpha_1) \neq 0$. Then $\pi_i(\rho) = \pi_i(\alpha_1 \star \beta_1) = \pi_i(\alpha_1)\pi_i(\beta_1)$ is a nonzero polynomial of degree 1 over L (in the variable $\pi_i(\beta_1)$), which occurs with probability at most $1/p^r$ according to Lemma 10. □

Proposition 8. *(Informal) Suppose that the shares* $\{[s_{j,i}]\}_{j=1,i=1}^{\ell,\kappa/h}$ *are authenticated, and let* P_u, P_v *be two players, where* P_v *is honest. If* P_u *announces potentially incorrect shares* $s'_{j,i,u} = s_{j,i,u} + \delta_{j,i,u}$ *and a potentially incorrect tag* $\tau' = \tau + \Delta$*, then the check* $\tau' \overset{?}{=} \boldsymbol{\mu} \odot \boldsymbol{\sigma}' + \nu$ *will succeed with probability at most* $\frac{1}{p^\kappa}$*.*

Proof. The errors $\delta_{j,i,u}$ translate into an error vector $\boldsymbol{\delta} \in (A_L)^\ell$ such that the check is performed on $\boldsymbol{\sigma}' = \boldsymbol{\sigma} + \boldsymbol{\delta}$. Furthermore, $\boldsymbol{\delta} = 0$ if and only if $\delta_{j,i,u} = 0$ for all $i \in \{1, \ldots, \kappa/h\}$ and $j \in \{1, \ldots, \ell\}$, so checking that the shares announced by P_u are correct amounts to checking that $\boldsymbol{\delta} = 0$.

It is easy to see that the check passes if and only if $\Delta = \boldsymbol{\mu} \odot \boldsymbol{\delta} + \nu$. Invoking Lemma 8 completes the proof. □

We conclude that once the tags are in place, these can be used to prevent corrupt parties to lie about their shares whenever some fault localization is required at the dispute control layer. We refer the reader to [5] for the details about how these tags are exactly used.

[16] The statement is incomplete since we are deliberately omitting many details like the dispute control layer, which determines which parties should get which type of tags, or how the keys are reused. We refer to [5] for these details.

Computation of the tags. In the previous paragraphs we showed that the tags, once computed and distributed, provide the required authentication properties. However, we did not deal with the way that these tags are computed. An important contribution of [5] was showing an efficient method for the computation of these tags, which saves in communication and that is crucial for the overall efficiency.

At a very high level, their method works as follows: First, observe that the task of computing the tags can be seen as a two-party protocol between party P_u and party P_v, where P_u inputs the share vector $\boldsymbol{\sigma}$, P_v inputs the keys $\boldsymbol{\mu} \in (A_S)^\ell$, $\nu \in A_S$, and P_u gets the output τ. The idea is to use a "Mini-MPC" protocol for this computation, but to ensure efficiency of the whole protocol distributing the inputs must be done with little communication. This is where the concept of twisted shares defined in Section 6.2 comes into play: one of the inputs, $\boldsymbol{\sigma}$, is actually a share, and therefore it is already "shared". We discuss this idea in a bit more detail in what follows, but first we begin with the crucial property of twisted shares that motivates their consideration in a first place.

Lemma 9. *Let $R = \mathsf{GR}(p^k, h)$, let $x, y \in R$ and suppose they are shared as $[x] = (x_1, \ldots, x_n) \in A^n$, $\lceil y \rfloor^i = (y_1, \ldots, y_n) \in A^n$. Then $[x] * \lceil y \rfloor^i = \langle\!\langle x_i \star y \rangle\!\rangle$. Furthermore, an analogous property holds for the LSSS obtained by extending to a Galois ring extension L.*

Proof. By definition, $\lceil y \rfloor^i$ can be seen as $[0]$. Then, using Proposition 5, we see that $[x] * \lceil y \rfloor^i = [x] * [0] = \langle 0 \rangle$. Furthermore, the i-th entry of this vector is $x_i \star y_i = x_i \star y$, which concludes the proof of the lemma. □

With this lemma in hand we can sketch the Mini-MPC protocol that the parties P_u, P_v use to compute the tags. First, let us assume for simplicity that $\ell = 1$ and that $R = S$, so the MAC is simply $\tau = \mu \star \sigma + \nu \in A$. Let $[s]$ be such that its u-th share is σ (recall that the tags are used to authenticate shares, so σ is a share of some secret). The protocol, at a high level, proceeds as follows:

1. P_v samples $\mu, \nu \in A$.
2. P_v distributes twisted shares of μ and double twisted shares of ν, i.e., $\lceil \mu \rfloor^u, \langle \nu \rangle^u$.
3. The parties compute $[s] * \lceil \mu \rfloor^u + \langle \nu \rangle^u$, which by Lemma 9 equals $\langle\!\langle \sigma \star \mu + \nu \rangle\!\rangle$.
4. The parties send these shares to P_u for reconstruction.
5. The correctness of the tags is verified via standard cut-and-choose techniques.

We refer the reader to protocol $\mathsf{TagComp}$ in [5] for the full details of the protocol to compute the tags. We remark that the core aspects of this protocol that depend on working over a field have been already addressed above, and the rest of the protocol translates directly to our setting.

Complexity analysis. With the due modifications the resulting $\mathsf{TagComp}$ protocol over R has a communication complexity of $O(k \cdot \log_2(p) \cdot (m \cdot n \cdot h + n^5 \cdot \kappa))$ for computing the tags in one single segment. Since $m = O(|C|)/n^2$, multiplying by the n^2 segments yields $O(k \cdot \log_2(p) \cdot (|C| \cdot n \cdot h + n^7 \cdot \kappa))$

6.6 Batched Triple Sacrifice

The task here is to compute the $M = O(|C|)$ multiplication triples necessary for the execution of online phase. Computing them can be done in a similar way as in Section 5, but their correctness will not be guaranteed. As before, due to the dispute control layer, $m = M/n^2$ triples are checked in each segment. One of the key novelties of the BFO protocol is a technique for checking these triples with a complexity that is roughly $O(n\log(n)+\kappa)$ per triple.[17] This is achieved by dividing the m triples to be checked into batches of size $N = n^2$ each, developing a procedure that checks these N triples with complexity $O(N\cdot n\cdot\log(n)+n^2\cdot\kappa)$, which, by multiplying by the number of batches m/N, yields $O(m\cdot(n\cdot\log(n)+\kappa))$.

Before we adapt their protocol to our setting, we begin by revisiting their techniques over fields here. Consider a field $\mathbb{F}_q$ with at least $2N$ elements, where $N = n^2$, and let $x_1,\dots,x_{2N-1}$ be different points in $\mathbb{F}_q$. Suppose the parties have shares over this field $\{[\![a_i]\!],[\![b_i]\!],[\![c_i]\!]\}_{i=1}^N$ where c_i is supposed to be $a_i\cdot b_i$. The parties check their consistency as follows:

1. Define $f(X), g(X)\in\mathbb{F}_q[X]$ to be the polynomials of degree at most $N-1$ such that $f(x_k)=a_k$ and $g(x_k)=b_k$ for $k=1,\dots,N$.
2. The parties compute shares of $a_k := f(x_k)$ and $b_k := g(x_k)$ for $k=N+1,\dots,2N-1$ by taking an appropriate linear combination (over $\mathbb{F}_q$) of the shares $\{[\![a_k]\!]\}_{k=1}^N$ and $\{[\![b_k]\!]\}_{k=1}^N$, respectively.
3. Define $h(X)$ as the polynomial of degree at most $2N-2$ given by $h(X) = f(X)\cdot g(X)$, notice that it should be the case that $c_k = h(x_k)$ for $k=1,\dots,N$.
4. Use a passively secure multiplication protocol to compute (potentially incorrectly) $[\![c_k]\!] := [\![a_k]\!]\cdot[\![b_k]\!]$ for $k=N+1,\dots,2N-1$. Now the parties have shares of $2N-1$ points on the polynomial $h(X)$.
5. Sample a random $\sigma\in\mathbb{F}_{q^\kappa}$ and compute shares over $\mathbb{F}_{q^\kappa}$ of $f(\sigma), g(\sigma), h(\sigma)\in\mathbb{F}_{q^\kappa}$ by taking a linear combination over $\mathbb{F}_{q^\kappa}$ of $\{[\![a_k]\!]\}_{k=1}^N$, $\{[\![b_k]\!]\}_{k=1}^N$ and $\{[\![c_k]\!]\}_{k=1}^{2N-1}$, respectively.
6. Perform some check over these shares to verify that $f(\sigma)\cdot g(\sigma) = h(\sigma)$.

When extending the above protocol over rings there are several complications that appear. One immediate concern is the argument that shows that checking the polynomial equality $f(X)\cdot g(X) = h(X)$ can be done by evaluating a random point. To show this still holds, we invoke the following lemma from [1, Lemma 2].

Lemma 10. *Let $f\in R[X]$ polynomial of arbitrary degree $\ell>0$. Then $\Pr_{x\leftarrow R}[f(x)=0]\le\frac{\ell}{p^\kappa}$, where x is drawn uniformly from R.*

One issue that appears is that we do not necessarily have enough points $x_1,\dots,x_{2N-1}\in R$ for interpolation over our ring R. We fix this by using a Galois ring extension L of degree $O(\log(N)) = O(\log(n))$ for the interpolation, which introduces an overhead of $\log(n)$ in the multiplications $[\![c_k]\!] = [\![a_k]\!]\cdot[\![b_k]\!]$

[17] A simple optimization in [5] transforms this into $O(n\log(n))$ for the case in which $\kappa = \mathsf{poly}(n)$. This optimization also applies to our setting.

for $k = N+1, \dots, 2N-1$. We remark that this is the *only* place of the whole protocol where the $\log(n)$ overhead appears.

The final effect of this is that the complexity of the preprocessing phase becomes $O(|C| \cdot (n \cdot \log(n) + \kappa))$, which is not fully linear, but it is already better than the best protocol known for this setting [1], which has a complexity of $O(|C| \cdot n^2 \cdot \log(n))$.[18] Furthermore, our online phase is fully linear, i.e., $O(|C| \cdot n)$. This has an interpretation in practice: in the offline phase the communication per party increases logarithmically as the number of parties gets larger, but in the online phase, this communication remains constant. This supports the rationale of the offline/online paradigm: expensive computations can be pushed to a function-independent preprocessing phase, and in the online phase where the inputs and the function are actually instantitated, the computation is cheaper.

We describe our protocol for batched triple generation in Fig. 2. It is very similar to the corresponding protocol in [5] except that in our case we use properties of Galois rings to argue about the security of the construction. The security of our construction is argued below in proposition 9. It shows that if there is at least one triple that is incorrect then it will be detected in the final check with high probability.

Proposition 9. *Let* $\{([\![a_i]\!]_R, [\![b_i]\!]_R, [\![c_i]\!]_R)\}_{i=1}^N$ *be the triples inputted to Protocol* $\mathsf{BatchedTriples}$*, and suppose that* $c_i = a_i \cdot b_i + d_i$ *for* $i = 1, \dots, N$*. If the honest parties output* OK *at the end of the protocol, then* $d_i = 0$ *for all* i *with probability at least* $1 - \frac{1}{p^\kappa}$.

Thanks to the properties of Galois rings that we have exploited throughout the paper, the proof follows along the same lines as the corresponding proof in [5], and we will not replicate it here.

Complexity analysis. Similar to the analysis in the field case done at the beginning of this section, the complexity of checking the m triples in one segment using $\mathsf{BatchedTriples}$ is $O(k \cdot \log_2(p) \cdot m \cdot (n \cdot \log(N) \cdot h + \kappa))$. By multiplying by the number of segments n^2, and recalling that $N = n^2$ and $m = O(|C|)/n^2$, we obtain $O(k \cdot \log_2(p) \cdot |C| \cdot (n \cdot \log(n) \cdot h + \kappa))$. Furthermore, the optimization in [5] of using $N = n^{2+c}$ where $\kappa(n) = O(n^c)$ applies also in our case and results in a complexity of $O(k \cdot \log_2(p) \cdot |C| \cdot n \cdot \log(n) \cdot h)$.

Remark 3. The $\log(n)$ overhead we have in the preprocessing appears in a very specific stage, and we can even remove it assuming a functionality that produces additive shares of matrix outer products efficiently.

[18] We notice, however, that the extension of Shamir secret sharing to R from [1] is likely to be compatible with the BFO protocol using some of the ideas introduced in our work. The resulting protocol would have the same offline complexity as our construction, but the online complexity would be $O(|C| n \log(n))$, unlike ours which is $O(|C| n)$. On the other hand, the threshold would be maximal.

BatchedTriples

Input: N potentially incorrect triples $\{([\![a_i]\!]_R, [\![b_i]\!]_R, [\![c_i]\!]_R)\}_{i=1}^N$.

– Let L be a Galois-ring extension of $\mathbb{Z}/p^k\mathbb{Z}$ of degree $\log_p(2N)$, and let $\chi_1, \dots, \chi_{2N-1} \in L$ be an exceptional sequence. We may think of S as a Galois-ring extension of L of degree $\kappa/\log_p(2N)$.

The parties execute the following in order to check the correctness of these triples:

1. Define $f(X), g(X) \in L[X]$ as the polynomials of degree at most $N-1$ such that $f(\chi_k) = a_k$ and $g(\chi_k) = b_k$ for $k = 1, \dots, N$.
2. The parties compute shares of $a_k := f(\chi_k) \in L$ and $b_k := g(\chi_k) \in L$ for $k = N+1, \dots, 2N-1$ by taking an appropriate linear combination $[\![a_k]\!]_L = \sum_{j=1}^N \lambda_j \cdot [\![a_j]\!]_R$ and $[\![b_k]\!]_L = \sum_{j=1}^N \lambda_j \cdot [\![b_j]\!]_R$, where $\lambda_j \in L$.
3. Define $h(X)$ as the polynomial of degree at most $2N-2$ given by $h(X) = f(X) \cdot g(X)$. Notice that it should be the case that $c_k = h(\chi_k)$ for $k = 1, \dots, N$.
4. For $i = N+1, \dots, 2N-1$ use the passively secure multiplication protocol to compute (possibly incorrect) shares $[\![a_i \cdot b_i]\!]_L = [\![a_i]\!]_L \cdot [\![b_i]\!]_L$.
5. Sample a random $\sigma \in S$ and compute shares of $f(\sigma), g(\sigma) \in S$ as $[\![f(\sigma)]\!]_S = \sum_{j=1}^N \mu_j \cdot [\![a_j]\!]_R$ and $[\![g(\sigma)]\!]_S = \sum_{j=1}^N \mu_j \cdot [\![b_j]\!]_R$ where $\mu_j \in S$.
6. Compute shares $[\![h(\sigma)]\!]_S = \sum_{i=1}^{2N-1} \omega_i \cdot [\![a_i \cdot b_i]\!]_L$, where $\omega_i \in S$.
7. The parties execute a *sacrifice* step in order to check the correctness of $([\![f(\sigma)]\!]_S, [\![g(\sigma)]\!]_S, [\![h(\sigma)]\!]_S)$:[a]
   (a) Sample a potentially incorrect triple $([\![\alpha]\!]_S, [\![\beta]\!]_S, [\![\gamma = \alpha \cdot \beta]\!]_S)$.
   (b) Sample a random value $\lambda \in S$.
   (c) Open sequentially $\alpha' = [\![\alpha]\!]_S + \lambda \cdot [\![f(\sigma)]\!]_S$ and $\nu = [\![\gamma]\!]_S + \gamma \cdot [\![h(\sigma)]\!]_S - \alpha' \cdot [\![g(\sigma)]\!]_S$.
   (d) If $v \neq 0$ then proceed to the fault localization phase as in [5]. Otherwise output OK.

[a] This corresponds to protocol SingleVerify in [5].

Fig. 2. Protocol for checking the correctness of several triples

Optimizing the batch triple verification. We can use the tools we have developed to further optimize our triple check procedure by adapting the more recent protocol of [21]. Their batch check protocol builds on top of the one we use from [5], and also makes use of polynomial interpolation, which as we have shown extends to Galois rings. This would lead to a more efficient protocol.

6.7 Putting the Pieces Together

Using the building blocks described in previous sections, we obtain a protocol over $R = \mathsf{GR}(p^k, h)$ whose offline phase has a total communication complexity of $O(k \cdot \log p \cdot (h \cdot |C| \cdot n \cdot \log(n) + n^7 \cdot \kappa))$. The online phase, which follows the exact same template as in [5, Section 3.4], has a total communication complexity of $O(k \cdot \log p \cdot h \cdot |C| \cdot n)$. This proves Theorem 10.

6.8 Linear-Complexity Preprocessing in the $\mathcal{F}_{\mathsf{OuterProduct}}$ Model

Finally, we would like to discuss an optimization to our base protocol. Unfortunately, our protocol in Section 6 does not achieve $O(|C|n)$ complexity in the preprocessing phase due to the computation of the N products in step 4 of our batched triple sacrifice, which happen over a Galois ring extension L of degree $\log(n)$. It turns out that if we assume an outer product functionality (to be defined below) we can obtain linear communication complexity by exchanging the N products over L with N^2 correlated products over R. The outer product functionality, denoted by $\mathcal{F}_{\mathsf{OuterProduct}}$, produces random triples $(\llbracket \boldsymbol{a} \rrbracket, \llbracket \boldsymbol{b} \rrbracket, \llbracket \boldsymbol{a} \cdot \boldsymbol{b}^\intercal \rrbracket)$, where $\boldsymbol{a}$ and $\boldsymbol{b}$ are (column) vectors over R of dimension N.

The reason why such functionality helps is the following. Suppose the parties have shares $\llbracket \mathbf{x} \rrbracket, \llbracket \boldsymbol{y} \rrbracket$, and the goal is to compute shares $\llbracket \mathbf{x} \cdot \boldsymbol{y}^\intercal \rrbracket$. Given a triple $(\llbracket \boldsymbol{a} \rrbracket, \llbracket \boldsymbol{b} \rrbracket, \llbracket \boldsymbol{a} \cdot \boldsymbol{b}^\intercal \rrbracket)$ produced by $\mathcal{F}_{\mathsf{OuterProduct}}$, this can be done by opening $\boldsymbol{d} = \llbracket \mathbf{x} \rrbracket - \llbracket \boldsymbol{a} \rrbracket$ and $\boldsymbol{e} = \llbracket \boldsymbol{y} \rrbracket - \llbracket \boldsymbol{b} \rrbracket$, and computing $\llbracket \mathbf{x} \cdot \boldsymbol{y}^\intercal \rrbracket = \boldsymbol{d} \cdot \llbracket \boldsymbol{b}^\intercal \rrbracket + \llbracket \boldsymbol{a} \rrbracket \cdot \boldsymbol{e}^\intercal + \llbracket \boldsymbol{a} \cdot \boldsymbol{b}^\intercal \rrbracket + \boldsymbol{d} \cdot \boldsymbol{e}^\intercal$. We denote this procedure by $\mathsf{OuterProduct}$. Notice that the communication complexity is $O(N)$, compared to the $O(N^2)$ one would get by computing each entry of $\llbracket \mathbf{x} \cdot \boldsymbol{y}^\intercal \rrbracket$ directly.

Given the procedure above, we modify steps 4, 5 and 6 of the protocol in Fig. 2 as follows:

4. Call $\mathsf{OuterProduct}$ on inputs $\{\llbracket a_i \rrbracket_R\}_{i=1}^N$ and $\{\llbracket b_i \rrbracket_R\}_{i=1}^N$ to get (potentially incorrect) shares $\{\llbracket a_i \cdot b_j \rrbracket_R\}_{i,j=1}^N$.
5. Sample a random $\sigma \in S$ and compute shares of $f(\sigma), g(\sigma) \in S$ as $\llbracket f(\sigma) \rrbracket_S = \sum_{j=1}^N \mu_j \cdot \llbracket a_j \rrbracket_R$ and $\llbracket g(\sigma) \rrbracket_S = \sum_{j=1}^N \mu_j \cdot \llbracket b_j \rrbracket_R$ where $\mu_j \in S$.
6. Observe that

$$h(\sigma) = \sum_{k=1}^{2N-1} \omega_k \cdot f(\chi_k) \cdot g(\chi_k) = \sum_{k=1}^{N} \omega_k \cdot a_k \cdot b_k + \sum_{k=N+1}^{2N-1} \omega_k \cdot \left(\sum_{i=1}^{N} \lambda_i \cdot a_i \right) \cdot \left(\sum_{j=1}^{N} \lambda_j \cdot b_j \right).$$

Therefore, the parties can compute

$$[\![h(\sigma)]\!]_S = \sum_{k=1}^{N} \omega_k \cdot [\![a_k]\!]_R \cdot [\![b_k]\!]_R + \sum_{i=1}^{N}\sum_{j=1}^{N} \eta_{ij} \cdot [\![a_i]\!]_R \cdot [\![a_j]\!]_R,$$

for some coefficients $\eta_{ij} \in S$.

We can see that with these modifications, the complexity becomes $O(N)$, which, when plugged into the general triple check, gives an overall complexity of $O(|C| \cdot n)$. However, this requires access to the $\mathcal{F}_{\mathsf{OuterProduct}}$ functionality, which does not seem straightforward to instantiate. For instance, it is not hard to see that this functionality cannot be instantiated with a communication complexity that is less than $O(N^2)$ if we aim for information-theoretic security. This has to do with the fact that the input has $O(N \cdot n)$ bits of entropy (from the point of view of the adversary), and the functionality requires the output to have $O(N^2 \cdot n)$ bits of entropy. Hence, any instantiation of this functionality must lie in the computational domain.

We stress that this verification step is the only part of the preprocessing phase where linearity is broken. Hence, we see our extension here to "fully-linear" preprocessing only as an indication of where *exactly* the main bottleneck lies.

7 Conclusions and Future Work

Our work shows that results from coding theory over fields can be leveraged to obtain corresponding results over the more general Galois rings, which include as a particular case the practically relevant ring $\mathbb{Z}/2^k\mathbb{Z}$. Although not all properties automatically lift (e.g., multiplicativity), we presented techniques to overcome these issues and still get meaningful coding-theoretic tools over Galois rings, that can be applied to MPC.

We showed that information-theoretic honest-majority MPC over rings which scales well with the number of parties is possible. Our protocols have linear communication complexity, except for the offline phase of our protocol with guaranteed output delivery from Section 6, which has a $\log(n)$ overhead. The complexity can be further reduced by combining our results with the work of [20].

Finally, like in [5], the communication complexity of our construction remains linear if the circuit is not too narrow. This restriction was removed in [20] for the case of $t < n/3$, and then in [21] for the case of $t < n/2$. As we mentioned in Section 6, the techniques from that paper can also be adapted to Galois rings.

Acknowledgements

The authors thank Gabriele Spini for helpful discussions in the early stages of this research project. This work has been supported by the European Union Horizon 2020 research and innovation programme under grant agreements No. 74079 (ALGSTRONGCRYPTO) and No. 669255 (MPCPRO), and by an SJTU-Huawei project.

References

1. M. Abspoel, R. Cramer, I. Damgård, D. Escudero, and C. Yuan. Efficient information-theoretic secure multiparty computation over $\mathbb{Z}/p^k\mathbb{Z}$ via Galois rings. In D. Hofheinz and A. Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 471–501. Springer, Heidelberg, Dec. 2019.
2. M. Abspoel, A. Dalskov, D. Escudero, and A. Nof. An efficient passive-to-active compiler for honest-majority MPC over rings. Cryptology ePrint Archive, Report 2019/1298, 2019. https://eprint.iacr.org/2019/1298.
3. Z. Beerliová-Trubíniová and M. Hirt. Efficient multi-party computation with dispute control. In S. Halevi and T. Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 305–328. Springer, Heidelberg, Mar. 2006.
4. Z. Beerliová-Trubíniová and M. Hirt. Perfectly-secure MPC with linear communication complexity. In R. Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 213–230. Springer, Heidelberg, Mar. 2008.
5. E. Ben-Sasson, S. Fehr, and R. Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 663–680. Springer, Heidelberg, Aug. 2012.
6. D. Boneh, E. Boyle, H. Corrigan-Gibbs, N. Gilboa, and Y. Ishai. Zero-knowledge proofs on secret-shared data via fully linear PCPs. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 67–97. Springer, Heidelberg, Aug. 2019.
7. E. Boyle, N. Gilboa, Y. Ishai, and A. Nof. Practical fully secure three-party computation via sublinear distributed zero-knowledge proofs. In L. Cavallaro, J. Kinder, X. Wang, and J. Katz, editors, *ACM CCS 2019*, pages 869–886. ACM Press, Nov. 2019.
8. I. Cascudo, R. Cramer, C. Xing, and C. Yuan. Amortized complexity of information-theoretically secure MPC revisited. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 395–426. Springer, Heidelberg, Aug. 2018.
9. H. Chen, R. Cramer, S. Goldwasser, R. de Haan, and V. Vaikuntanathan. Secure computation from random error correcting codes. In M. Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 291–310. Springer, Heidelberg, May 2007.
10. K. Chida, D. Genkin, K. Hamada, D. Ikarashi, R. Kikuchi, Y. Lindell, and A. Nof. Fast large-scale honest-majority MPC for malicious adversaries. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 34–64. Springer, Heidelberg, Aug. 2018.
11. R. Cramer, I. Damgård, D. Escudero, P. Scholl, and C. Xing. SPD $\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for dishonest majority. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 769–798. Springer, Heidelberg, Aug. 2018.
12. R. Cramer, I. Damgård, and U. M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 316–334. Springer, Heidelberg, May 2000.
13. R. Cramer, S. Fehr, Y. Ishai, and E. Kushilevitz. Efficient multi-party computation over rings. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 596–613. Springer, Heidelberg, May 2003.

14. R. Cramer, M. Rambaud, and C. Xing. Asymptotically-good arithmetic secret sharing over $\mathbb{Z}/p^k\mathbb{Z}$ with strong multiplication and its applications to efficient MPC. *IACR Cryptology ePrint Archive*, 2019:832, 2019.
15. I. Damgård, K. G. Larsen, and J. B. Nielsen. Communication lower bounds for statistically secure MPC, with or without preprocessing. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 61–84. Springer, Heidelberg, Aug. 2019.
16. I. Damgård and J. B. Nielsen. Scalable and unconditionally secure multiparty computation. In A. Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 572–590. Springer, Heidelberg, Aug. 2007.
17. I. Damgård, D. Escudero, T. Frederiksen, M. Keller, P. Scholl, and N. Volgushev. New primitives for actively-secure MPC over rings with applications to private machine learning. In *2019 2019 IEEE Symposium on Security and Privacy (SP)*, pages 1325–1343, Los Alamitos, CA, USA, may 2019. IEEE Computer Society.
18. A. Garcia and H. Stichtenoth. A tower of Artin-Schreier extensions of function fields attaining the Drinfeld-Vladut bound. *Inventiones mathematicae*, 121(1):211–222, Dec 1995.
19. D. Genkin, Y. Ishai, M. Prabhakaran, A. Sahai, and E. Tromer. Circuits resilient to additive attacks with applications to secure computation. In D. B. Shmoys, editor, *46th ACM STOC*, pages 495–504. ACM Press, May / June 2014.
20. V. Goyal, Y. Liu, and Y. Song. Communication-efficient unconditional MPC with guaranteed output delivery. In A. Boldyreva and D. Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 85–114. Springer, Heidelberg, Aug. 2019.
21. V. Goyal, Y. Song, and C. Zhu. Guaranteed output delivery comes free in honest majority MPC. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 618–646. Springer, Heidelberg, Aug. 2020.
22. L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. In *Concurrency: the Works of Leslie Lamport*, pages 203–226. 2019.
23. Y. Lindell and A. Nof. A framework for constructing fast MPC over arithmetic circuits with malicious adversaries and an honest-majority. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *ACM CCS 2017*, pages 259–276. ACM Press, Oct. / Nov. 2017.
24. J. L. Massey. Some applications of coding theory in cryptography. In P. F. (ed.), editor, *Codes and Ciphers, Cryptography and Coding IV*, pages 33–47, Formara Lt, Esses, England, 1995.
25. P. S. Nordholt and M. Veeningen. Minimising communication in honest-majority MPC by batchwise multiplication verification. In B. Preneel and F. Vercauteren, editors, *ACNS 18*, volume 10892 of *LNCS*, pages 321–339. Springer, Heidelberg, July 2018.
26. H. Stichtenoth. Transitive and self-dual codes attaining the Tsfasman-Vlăduţ-Zink bound. *IEEE Trans. Inf. Theor.*, 52(5):2218–2224, May 2006.
27. H. Stichtenoth. *Algebraic Function Fields and Codes*. Springer Publishing Company, Incorporated, 2nd edition, 2008.