Improved Single-Round Secure Multiplication Using Regenerating Codes

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

[1] CWI, Amsterdam
[2] Leiden University
[3] J.P. Morgan AI Research*
[4] Aarhus University
[5] School of Electronic Information and Electric Engineering, Shanghai Jiao Tong University

Abstract. In 2016, Guruswami and Wootters showed Shamir's secret-sharing scheme defined over an extension field has a regenerating property. Namely, we can compress each share to an element of the base field by applying a linear form, such that the secret is determined by a linear combination of the compressed shares. Immediately it seemed like an application to improve the complexity of unconditionally secure multiparty computation must be imminent; however, thus far, no result has been published.

We present the first application of regenerating codes to MPC, and show that its utility lies in reducing the number of rounds. Concretely, we present a protocol that obliviously evaluates a depth-d arithmetic circuit in $d + O(1)$ rounds, in the amortized setting of parallel evaluations, with $o(n^2)$ ring elements communicated per multiplication. Our protocol makes use of function-dependent preprocessing, and is secure against the maximal adversary corrupting $t < n/2$ parties. All existing approaches in this setting have complexity $\Omega(n^2)$.

Moreover, we extend some of the theory on regenerating codes to Galois rings. It was already known that the repair property of MDS codes over fields can be fully characterized in terms of its dual code. We show this characterization extends to linear codes over Galois rings, and use it to show the result of Guruswami and Wootters also holds true for Shamir's scheme over Galois rings.

1 Introduction

Secret-sharing is a technique that enables a given secret to be distributed into multiple values, called *shares*, in such a way that certain subsets of these do not leak anything about the secret, whereas certain other subsets completely determine the secret. These techniques are used in a wide range of areas such as

* Work done while Daniel Escudero was at Aarhus University

distributed storage, cloud computing and multiparty computation, and improving/expanding them in different directions has huge impact in a lot of different domains.

The theory of secret-sharing schemes shares a deep connection with the theory of error-correcting codes, with many results in one domain translating naturally to results in the other. In 2016, Guruswami and Wootters showed that in a certain parameter regime, Reed-Solomon codes have a regenerating property [13], which in terms of secret-sharing can be translated into reconstructing a secret using "incomplete" shares. In this context, we can illustrate this property with the following example.

Consider Shamir's secret-sharing scheme with n shares and t-privacy defined over the binary extension field $\mathbb{F}_{2^m}$, subject to the regime $t < n - 2^{m-1}$. Suppose we are in an interactive scenario, where n parties $P_1, \ldots, P_n$ are connected by pairwise communication channels, with each party having a distinct share. We know Shamir's scheme has $(t+1)$-reconstruction: the secret can be computed from any subset of $t+1$ shares. Therefore, if the parties wish to reconstruct the secret value towards one of the parties, say P_1, they can do so by having t other parties send their share, an element in $\mathbb{F}_{2^m}$, to P_1, resulting in $m \cdot t$ bits of communication.

However, it turns out that it suffices for *all* parties to send a *single bit* to P_1, reducing communication by a factor $mt/(n-1)$. To accomplish this, each party P_i applies an $\mathbb{F}_2$-linear *compression function* $\phi_i : \mathbb{F}_{2^m} \to \mathbb{F}_2$ to their share, each of which is chosen such that the n compressed shares jointly determine the secret.

As mentioned before, secret-sharing techniques play a key role in information-theoretic multiparty computation (MPC). In this context, there is a set of participants $P_1, \ldots, P_n$ who want to jointly compute a given function on private data without leaking anything but the output. At a high level, the standard approach consists of letting the parties hold the inputs in secret-shared form, and designing methods to obtain shares of every intermediate step of the computation, until shares of the desired output are reached. At this point, this secret-shared value is reconstructed towards the party who is intended to learn the given output.

Reconstructing secret-shared values is a crucial step in MPC protocols, as this does not only happen in the output phase when the result of the computation must be reconstructed: most MPC protocols require reconstruction of secret-shared values for every non-trivial operation like a multiplication. Given this, it might seem at first sight that the techniques developed by Guruswami and Wootters would immediately improve communication complexity for information-theoretic multiparty computation (MPC). However, so far, a concrete application has remained elusive.[6] There are a number of factors that play into this.

[6] When Mary Wootters presented this result to the community in an invited talk of the Beyond TCS workshop affiliated to CRYPTO 2018, she posed the question to the community of what its implications to MPC are. It generated a bit of a buzz, with several members of the community working on it even during the conference,

First, one general observation is that since the reduction in communication is proportional to m, the largest improvement is obtained when m is large. An implication is that since $0 < t < n - 2^{m-1}$, this also means the number of players must be large. Therefore, we restrict ourselves to asymptotic improvements only. In the following, we assume the adversary threshold t is at least linear in n.

Second, even though regenerating codes apply to large fields, the function that we wish to compute via MPC is typically expressed as a circuit over a finite field of small fixed size, such as $\mathbb{F}_2$. Efficient computation over $\mathbb{F}_{2^m}$ to evaluate circuits over $\mathbb{F}_2$ gives us an advantage in the amortized model, where we execute the same circuit many times in parallel with different inputs. In this model, we can obtain a lower amortized communication complexity (per circuit evaluation) by using reverse multiplication-friendly embeddings (RMFEs) [4].

Third, using compressed shares only improves communication for the reconstruction of a secret, and not for secret-sharing a value. This means that we cannot hope to easily improve the standard 2-round protocol for secure multiplication. In more detail, one way to securely multiply secret-shared values x and y is for the parties to consume an additional random secret-shared element r, and reconstruct their share of $\delta := xy - r$ towards one party, who subsequently broadcasts δ. If we assume this broadcast is cheap, the online cost of this secure multiplication is essentially the same as the reconstruction of one secret-sharing. However, as mentioned, regenerating codes will not help us optimize the generation of the random secret-shared r, so the overall protocol still requires $\Omega(n)$ bits of communication[7].

All this leaves only one setting in which we can meaningfully ask if regenerating codes can help: we can consider the tradeoff between the communication complexity and the number of rounds. The multiplication protocol we just considered has complexity $O(n)$ bits, which is asymptotically optimal [9], however, it uses two rounds.[8] It can be modified to use only *a single round*, where each player simply sends his share of δ to all players. But then the complexity increases to $\Theta(n^2)$. In fact, no currently known single-round protocol beats this bound, and we conjecture that it is optimal for the non-amortized setting. Note that in the amortized setting we can use packed secret sharing to get an improved single-round multiplication. However, this only works for a submaximal adversary $t < (1/2 - \varepsilon)n$.

Decreasing the round complexity of secure multiparty computation protocols is a well-motivated goal. Since information-theoretic MPC protocols typically evaluate a circuit gate by gate, they require a number of sequential interactions of

however no result has been published thus far. We remark that other applications to regenerating codes to, for example, leakage-resilience of secret-sharing schemes [3] or side-channel countermeasures have been proposed, but none of these study positive effects in MPC constructions [5].

[7] We could try to get around this using computationally secure pseudorandom secret sharing, but this requires an exponential number of keys in n.

[8] The communication of the king-based protocol is $O(n)$ *field elements* for the maximal adversary $n = 2t + 1$. By incorporating a constant-rate RMFE we can achieve a communication of $O(n)$ *bits*, which is asymptotically optimal [9].

at least the round complexity per multiplication times the circuit depth. When the network latency is high, such as in wide area networks, the number of rounds can become the dominant factor in the running time of the protocol. Furthermore, there are even scenarios where *single-round* multiplication is essential.[9] For example, the work of [6] introduces the concept of "fluid MPC protocols", where the set of compute parties changes from one round to the next, enabling secure computation in a dynamic setting such as blockchains. In that work, the authors present a protocol that heavily relies on secure multiplication in a single round, and they leave it as an open problem to obtain a fluid MPC protocol that requires less than $\Theta(n^2)$ bits of communication per multiplication.

Motivated by the above, the question we ask is: can regenerating codes help us to build a one-round secure multiplication protocol in the amortized setting (and for a maximal adversary) where the complexity is $o(n^2)$?

1.1 This Work

In a nutshell, we give an answer in the affirmative to the question above by presenting an MPC protocol that, by making use of regenerating codes, achieves a round complexity that is not only proportional to the multiplicative depth of the circuit, but is essentially equal to it, while keeping sub-quadratic communication complexity in the number of parties. Our protocol achieves statistical security against an active adversary corrupting a minority of the parties. Furthermore, our protocol and techniques are not restricted to finite fields only: they are set in the context of *Galois rings*, which are a natural generalization involving power-of-primes characteristics. All finite fields are Galois rings, but Galois rings also include non-field rings such as $\mathbb{Z}/p^k\mathbb{Z}$, that have gained popularity for MPC recently.

In order to obtain our results, we first show in Section 4, as a contribution on its own, that the repair property is equivalent to a condition on the dual code containing a particular subcode. This was already noted in [12] for MDS codes over fields, but we give an alternative proof that extends to arbitrary linear codes over Galois rings. From this characterization, we obtain a generalization of the result of [13], as we show that Shamir's scheme over Galois rings [1] also has the repair property.

As mentioned before, we show that the utility of regenerating codes in information-theoretic MPC lies in reducing the number of rounds, and we obtain the first application of regenerating codes in this domain. Answering the above question, we reduce the communication complexity of single-round secure multiplication from $O(n^2)$ to $O(n^2/\log(n))$ ring elements, while requiring only $d + O(1)$ rounds for a depth-d circuit.

Our results are obtained by using our extended characterization of regenerating codes over Galois rings in order to reconstruct secret-shared values efficiently,

[9] We call a single-round protocol one that only requires one round per multiplication layer in the circuit.

together with the standard approach for MPC based on preprocessed multiplication triples. Furthermore, in order to effectively make use of the large-degree Galois ring extension required for regenerating codes to exist, we employ RMFEs [4, 8] to compute several copies of the same circuit in parallel, achieving the desired communication complexity for each individual execution (i.e. after dividing by the total number of copies). Whether our results can also be achieved in the non-amortized setting, i.e. when only one execution is required, is left as an open problem. We conjecture, however, that this is not possible, and provide arguments for this in Section 5.

In a bit more detail, we first show in Section 4.1 that a direct application of our regenerating code characterization, together with standard multiplication protocols based on multiplication triples, yield a single-round multiplication protocol with non-trivial communication complexity, assuming that the degree of the Galois ring extension is large enough. Then, in Sections 4.2 and 4.3 we show how to leverage the simple protocol described in Section 4.1 in order to compute several copies of the same circuit in parallel, which yields our final result. This is achieved, as mentioned before, by making use of RMFEs to pack multiple elements of the base ring into a Galois ring extension element.

Unfortunately, this amortization step is far from trivial. Although, in [4], the authors present a compiler that takes any multiplication protocol over a field extension, and turns it into a multiplication protocol for several parallel multiplication over the corresponding base field, we cannot make use of these techniques directly to obtain our results. This is not only because these techniques are set over fields rather than Galois rings, which is possible to fix [8]. Instead, the main complication comes from the fact our protocol is highly constrained in that it must achieve secure multiplication in *one round*, and the RMFE-based protocol from [4] adds an extra round for each multiplication. This is due to the fact that the actual multiplication protocol needs to do more than just a multiplication in the large field; rather, the real goal of the protocol is to coordinate-wise multiply vectors of values in the small field, and this requires encoding them as elements in the large field, with a re-encoding step after every large field multiplication.

In order to overcome this issue, we show that we can use the same RMFE but encode values in a different manner, which allows us to fit the entire multiplication into a single round. This contribution can be of independent interest, given that it also removes the necessity of re-encoding the output of each multiplication in the original compiler of [4] (plus removing a necessary subdomain check in the input phase to ensure security against active adversaries). Works that make use of these techniques will also benefit from our encoding improvement.

Finally, since our protocol operates by opening secret-shared values using compressed shares, which do not offer any redundancy for detecting errors, consistency in the event of an active adversary may not be guaranteed during the execution of the protocol. In order to ensure that no cheating took place, a single check that aggregates all the multiplications performed during the protocol execution is run at the end, which results in unconditional security with abort. A subtle issue with this approach is that, allowing the parties to inadvertently continue with the execution of the protocol in spite of the adversary injecting errors in some of the multiplications, may harm privacy. To overcome this problem, we make use of multiplication triples in a different way than how it is done traditionally. This is achieved by employing a novel use of *function-dependent preprocessing*, which was already used in [2, 11] to improve the communication complexity of MPC protocols. To the best of our knowledge, ours constitutes the first work that identifies this technique to be beneficial for security purposes as well.

Our protocol assumes an honest majority $t < n/2$, and works for the maximal adversary $n = 2t + 1$. Our main result in terms of MPC is described in the theorem below.

Theorem 1. *There exists a family of protocols, indexed by the number of parties $n \to \infty$, that privately computes a depth-d arithmetic circuit over $\mathbb{Z}/p^k\mathbb{Z}$ with abort many times in parallel on different vectors of inputs in $d + O(1)$ rounds, and communicates $o(n^2 \log_2(p))$ bits per multiplication gate. The protocols make use of function-dependent preprocessing and are secure against an active adversary that can corrupt $t < n/2$ parties and can also abort the computation.*

We remark that the communication costs discussed above are asymptotic in terms of the number of multiplication gates of the underlying circuit. If there are M multiplication gates, the cost of evaluating these securely turns out to be $o(n^2 M)$, plus a term $\approx n^2$, which vanishes as $M \to \infty$.

1.2 Related Work

The task of minimizing the number of rounds in unconditionally-secure MPC has not received much attention, at least when compared to the task of minimizing communication. It is a long-standing open problem whether constant-round protocols that are efficient (i.e. have polynomial communication complexity in n for functionalities beyond NC^1) and enjoy information-theoretic security exist. In fact, in [14] a tight relation between this problem and a problem on private information retrieval, considered to be among the most intriguing open problems in complexity theory, was established. Given this, existing unconditionally secure MPC protocols have a number of rounds that grows proportionally to the multiplicative depth of the circuit. One of the first protocols in this direction is the BGW protocol [20], which presents an information-theoretically secure protocol that requires $d + O(1)$ rounds to securely compute a function with multiplicative depth d, but requires $\Omega(n^2)$ communication for each multiplication. More modern protocols with linear communication in the number of parties, such as the popular DN07 protocol [10], but at the expense of requiring $c \cdot d + O(1)$ rounds for some $c > 1$.

To the best of our knowledge, Galois rings have not been considered within the literature of regenerating codes, so there are no related works in this direction. Furthermore, as we stressed in previous paragraphs, applications of regenerating codes to MPC have been elusive until this point. However, even though regenerating codes have not been used for secret-sharing in MPC, there has been a big body of research that makes use of these techniques for secret-sharing in the context of distributed storage. For instance, [17] studies the problem of handling errors and erasures during the data-reconstruction and node-repair operations, and proposes explicit regenerating codes that resist errors and erasures, and show their optimality. Also, in [16] constructions of the so-called Minimum Bandwidth Regenerating (MBR) and Minimum Storage Regenerating (MSR) codes are presented, which are useful for distributed storage. We stress that in works along these lines, secret-sharing is used as a mean to store information, rather than as a method for secure computation, which is the setting we consider in our work.

Finally, we note that there are other constructions of regenerating codes on top of Reed-Solomon codes, which are follow-ups to the original work of Guruswami and Wootters, which is the one we use as a starting point in our work. For instance, in [18], Reed-Solomon codes achieving the so-called cut-set bound are introduced, which represent an improvement over the original construction in terms of savings when performing regeneration. We believe such works would be beneficial for MPC too by following a similar approach as the one we present in our work, and we leave it as future work to explore these potentially fruitful directions.

2 Preliminaries

Galois rings. All rings that we refer to are commutative and have a multiplicative identity 1. For a ring R and R-modules A, B, we denote by $\mathrm{Hom}_R(A, B)$ the R-module of R-linear maps from A to B.

A Galois ring R is a finite ring such that the set of zero divisors, with 0 added, forms a principal ideal generated by $p \cdot 1$ where $p \in \mathbb{Z}$ is prime. It is a local ring, whose maximal ideal is precisely the ideal (p) of zero divisors. R is isomorphic to the ring $(\mathbb{Z}/p^k\mathbb{Z})[X]/(h(X))$, where k is a positive integer and p is prime, and $h(X) \in (\mathbb{Z}/p^k\mathbb{Z})[X]$ is a monic polynomial such that its reduction modulo p is irreducible in $\mathbb{F}_p[X]$. Conversely, all rings of this form are Galois rings, and a choice of p, k and $m = \deg h(X)$ uniquely defines the Galois ring up to isomorphism, so that we may write $R = \mathrm{GR}(p^k, m)$. The kernel of the unique ring homomorphism $\mathbb{Z} \to R$ is the ideal $(p^k) \subset \mathbb{Z}$, hence the characteristic of R is $\mathrm{char}(R) = p^k$. All finite fields, as well as the rings $\mathbb{Z}/p^k\mathbb{Z}$, are Galois rings.

From the definition it is easy to see that every element of a Galois ring $R = (\mathbb{Z}/p^k\mathbb{Z})[X]/(h(X))$ can be written uniquely as a polynomial of degree at most $m - 1$ and coefficients in $\mathbb{Z}/p^k\mathbb{Z}$. However, another representation that we will find useful in this work is the following.

Theorem 2 (Theorem 14.8 in [19]). *Let $R = \mathrm{GR}(p^k, m)$. There exists $\xi \in R$ of order $p^m - 1$ such that every $c \in R$ can be written uniquely as $c = a_0 + a_1 p + \cdots + a_{k-1}p^{k-1}$, where $a_i \in \{0, 1, \xi, \xi^2, \ldots, \xi^{p^m-2}\}$ for $i = 0, \ldots, k-1$. Moreover, c is a unit if and only if $a_0 \neq 0$, and c is a zero divisor, or zero, if and only if $a_0 = 0$.*

Suppose we have a subring $S \subseteq R$. Then S is a Galois ring with $\mathrm{char}(R) = \mathrm{char}(S)$, and we call R/S an extension of Galois rings. If $R = \mathrm{GR}(p^k, m)$ and $S = \mathrm{GR}(p^k, n)$, then $n \mid m$. We call m/n the *degree* of the extension, and denote it $[R : S]$. Proofs of the above assertions and more details on Galois rings can be found in [19]. More details on Galois rings in the context of secret sharing and MPC can be found in [1].

Generalized Reed-Solomon codes. Let t, n be non-negative integers with $t < n$. We denote by $R[X]_{\leq t}$ ($R[X]_{<t}$) the free R-module of polynomials over R of degree at most (strictly less than) t. A sequence of elements $\alpha_1, \ldots, \alpha_n \in R$ is called exceptional if $\alpha_i - \alpha_j$ is a unit for each pair of distinct indices $i \neq j$. There exists an exceptional sequence in R of length p^m (e.g., lift each element of the residue field to R), and this is the maximum length. Given such an exceptional sequence, and a vector of units $(y_1, \ldots, y_n) \in (R^*)^n$, a generalized Reed-Solomon code over R of length n and rank $t+1$ is an R-submodule $C \subseteq R^n$ given by

$$C = \left\{ (y_1 f(\alpha_1), \ldots, y_n f(\alpha_n)) \;\middle|\; f \in R[X]_{\leq t} \right\}.$$

Reverse multiplication-friendly embeddings. Let ℓ be a positive integer and write $m := [R : S]$. We denote by S^ℓ the S-module of ℓ copies of S. It is also an S-algebra with respect to the coordinatewise product $*$. An (ℓ, m)-*reverse multiplication-friendly embedding (RMFE) for* R/S is a pair of S-linear maps $\phi : S^\ell \to R$, $\psi : R \to S^\ell$, such that

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

for all $\mathbf{x}, \mathbf{y} \in S^\ell$.

We are particularly interested in RMFEs for $R/(\mathbb{Z}/p^k\mathbb{Z})$, since they allow us to evaluate parallel circuits over $\mathbb{Z}/p^k\mathbb{Z}$ using MPC over R [4]. Such RMFEs exist, even with the property of being asymptotically good (i.e., with the rate ℓ/m tending to a positive constant). This was shown in the following theorem from [8, Theorem 22].

Theorem 3. *There exists a family of (ℓ, m)-RMFEs, indexed by $m \to \infty$, for the Galois ring extensions* $\mathrm{GR}(p^k, m)/(\mathbb{Z}/p^k\mathbb{Z})$ *with* $\ell = \Omega(m)$.

Security model. For the security proofs of our protocol we make use of the UC model for multiparty computation. Details can be found in [7]. Also, we assume that whenever an honest party aborts, all the honest parties abort. This can be assumed without loss of generality given that we assume a broadcast channel, so the abort signals can be transmitted through this medium.

3 Regenerating Codes over Galois Rings

Let R/S be an extension of Galois rings of characteristic p^k, with $S = \mathrm{GR}(p^k, \ell)$ and $R = \mathrm{GR}(p^k, m \cdot \ell)$. Let n be a positive integer, and let $C \subseteq R^{n+1}$ be an

R-submodule with coordinates indexed by $0, 1, \ldots, n$. For each index i we denote the projection map onto the i-th coordinate by $\pi_i : C \to R$. We say C is a *regenerating code* if it has the following *repair property*.[10]

Definition 1. *An R-submodule $C \subseteq R^{n+1}$ has* linear repair over S of the 0-coordinate *if for each index $i > 0$ there exists an S-linear map $\phi_i : R \to S$ and a scalar $z_i \in R$, such that for each element $(x_0, x_1, \ldots, x_n) \in C$ it holds that $x_0 = \sum_{i=1}^{n} \phi_i(x_i) \cdot z_i$.*

We now show that the repair property can be fully characterized in terms of the dual code $C^\perp$. This was already shown in [12] for MDS codes over fields, but we show it in our setting of 1-dimensional repair of the 0-coordinate, and demonstrate it extends to arbitrary linear codes over Galois rings.

Theorem 4. *Let $C \subseteq R^{n+1}$ be an R-submodule. Then C has linear repair over S of the 0-coordinate if and only if there exists an S-submodule $D_0 \subseteq C^\perp$ of the dual code, with the following properties.*

1. *$\pi_0(D_0) = R$*
2. *For each index $i > 0$ there is some integer j with $0 \leq j \leq k$, such that $\pi_i(D_0) \cong p^j S$ as S-modules.*

From this characterization, we will below easily derive a generalization of a result of [13], namely that Reed-Solomon codes over *Galois rings* have linear repair. To prove the theorem we use two general lemmas.

Lemma 1. *Let $f : R \to S$ be a surjective S-linear map. For each $\alpha \in R$, let $f_\alpha : R \to S$ denote the S-linear map given by $x \mapsto f(\alpha x)$. Then, the map*

$$\begin{aligned} R &\longrightarrow \mathrm{Hom}_S(R, S) \\ \alpha &\longmapsto f_\alpha \end{aligned}$$

is an S-module isomorphism.

Proof. We observe the map is S-linear. Since $R \cong S^{[R:S]}$ as S-modules, we have that R and $\mathrm{Hom}_S(R, S)$ are two finite sets of the same cardinality. Therefore, it suffices to show injectivity.

Let $\alpha \in R$ be nonzero. By surjectivity of f, there exists $w \in R$ such that $f(w) = 1$. It must hold that w is a unit, otherwise $p^{k-1} = p^{k-1} f(w) = f(p^{k-1} w) = f(0) = 0$. Write $\alpha = p^t u$, where t is an integer with $0 \leq t < k$ and $u \in R^*$ is a unit. For $x := u^{-1} w$ we have that $f_\alpha(x) = f(\alpha x) = f(p^t w) = p^t \neq 0$, which shows that f_α is not the zero map. $\square$

Lemma 2. *Let $f : R \to S$ be a surjective S-linear map. Let $x, y \in R$. Then $x = y$ if and only if, for all $\gamma \in R$, we have $f(\gamma x) = f(\gamma y)$.*

[10] We only regard 1-dimensional repair of the 0-th coordinate, since we specifically target applications to MPC. In the literature on regenerating codes, the definition typically includes all coordinates and allows for larger messages to be sent.

Proof. For an arbitrary $\gamma \in R$, we have $f(\gamma x) = f(\gamma y)$ if and only if $f((x-y)\gamma) = 0$. The latter holds for all $\gamma \in R$ if and only if f_{x-y} is the zero map, which by Lemma 1 holds if and only if $x - y = 0$. $\square$

With these two lemmas at hand, we are now ready to give a proof of Theorem 4.

Proof (of Theorem 4). Let $f : R \to S$ be any surjective S-linear map (for example, choose an S-basis of R and project onto the first coordinate). Assume C has linear repair, i.e. there exist maps $\phi_1, \ldots, \phi_n : R \to S$ and elements $z_1, \ldots, z_n \in R$ such that for all $\mathbf{x} = (x_0, x_1, \ldots, x_n) \in C$ we have $x_0 = \sum_{i=1}^n \phi_i(x_i) z_i$. By Lemma 2 this equality holds if and only if for all $g \in R$ we have

$$f(gx_0) = f\left(g \sum_{i=1}^{n} \phi_i(x_i) z_i\right) = \sum_{i=1}^{n} f(g\phi_i(x_i)z_i) = \sum_{i=1}^{n} f(gz_i)\phi_i(x_i).$$

Using Lemma 1, we write each $\phi_i(x_i)$ as $f(\beta_i x_i)$, for some $\beta_1, \ldots, \beta_n \in R$, and obtain

$$f(gx_0) = \sum_{i=1}^{n} f(gz_i) f(\beta_i x_i) = \sum_{i=1}^{n} f(f(gz_i)\beta_i x_i) = f\left(\sum_{i=1}^{n} f(gz_i)\beta_i x_i\right).$$

By R-linearity of C we may replace $\mathbf{x}$ by $\gamma\mathbf{x}$ for arbitrary $\gamma \in R$, therefore we may apply Lemma 2 and see equality holds without application of f. Equivalently, the vector $(-g, f(gz_1)\beta_1, \ldots, f(gz_n)\beta_n)$ is in the dual $C^\perp$.

Let D_0 denote the collection of these vectors where g varies over R, and note that $\pi_0(D_0) = -R = R$. Now let $i > 0$ be any index, and consider the projection

$$\pi_i(D_0) = \left\{ f(gz_i)\beta_i \;\middle|\; g \in R \right\} = \left\{ f(gz_i) \;\middle|\; g \in R \right\} \beta_i.$$

We have that $\{f(gz_i) \mid g \in R\}$ is an S-submodule of S, hence it is an ideal of S, and therefore equal to $p^j S$ for some nonnegative integer j. We can write $\beta_i = p^{j'} u$, where $u \in R^*$ is a unit and j' is some nonnegative integer. Multiplication by u gives an S-module isomorphism $p^{j'} S \cong p^{j'} S u = S\beta_i$. We conclude $\pi_i(D_0) \cong p^{j+j'} S$, and remark that if $j + j' \geq k$ this is equal to $p^k S = 0$, thus proving the forward direction of the theorem.

For the converse, assume we have $D_0 \subseteq C^\perp$ as in the theorem. From the second condition of D_0, we know there exist $\beta_1, \ldots, \beta_n \in R$ such that $\pi_i(D_0) = S\beta_i$ for each index $i > 0$. Now, we choose an S-basis of R, say $b_1, \ldots, b_m \in R$. By the first condition on D_0, we have that for each b_j there exist $\lambda_1^{(b_j)}, \ldots, \lambda_n^{(b_j)} \in S$ such that $\left(-b_j, \lambda_1^{(b_j)}\beta_1, \ldots, \lambda_n^{(b_j)}\beta_n\right) \in D_0$. For each $g \in R$ we may write $g = \sum_{j=1}^m g_j b_j$, hence by S-linearity of D_0 there exists a vector in D_0 whose zeroth component is $-g$ and for each index $i > 0$ its i-th component is $\left(\sum_{j=1}^m g_j \lambda_i^{(b_j)}\right)\beta_i$. Applying Lemma 1 there exist fixed z_i for each index $i > 0$ such that for all $g \in R$ we have $f(gz_i) = \sum_{j=1}^m g_j \lambda_i^{(b_j)}$. We now have that for each $g \in R$, there exists a vector

$$(-g, f(gz_1)\beta_1, \ldots, f(gz_n)\beta_n) \in C^\perp.$$

We can follow the steps of the proof above in reverse direction, and writing $\phi_i(x_i) := f(\beta_i x_i)$ for each index $i > 0$, we conclude for each $(x_0, \dots, x_n) \in C$ we have that $x_0 = \sum_{i=1}^{n} \phi_i(x_i) z_i$. $\square$

We now use our characterization of the repair property to show the result of [13] generalized to Galois rings. Namely, a generalized Reed-Solomon code defined over R has linear repair over S of the coordinate corresponding to the evaluation point 0. To show this we make use of the generalized trace function corresponding to the extension of Galois rings R/S.

An important lemma we will make use of is the following, which is a natural generalization of the equivalent result over fields. A proof of this result can be found, for example, in [15].

Lemma 3. *Let $\alpha_0, \alpha_1, \dots, \alpha_n \in R$ be an exceptional sequence. Let $\mathbf{y} = (y_0, \dots, y_n) \in (R^*)^{n+1}$ be a vector of units, and let C be the generalized Reed-Solomon code of rank $t+1$ and length $n+1$ given by*

$$C = \left\{ (y_0 f(\alpha_0), y_1 f(\alpha_1), \dots, y_n f(\alpha_n)) \;\middle|\; f \in R[X]_{\leq t} \right\}.$$

Then the dual of C is given by the code of length $n+1$ and rank $n-t$ given by

$$C^{\perp} = \left\{ (w_0 f(\alpha_0), w_1 f(\alpha_1), \dots, w_n f(\alpha_n)) \;\middle|\; f \in R[X]_{< n-t} \right\},$$

where $w_i = \left(y_i \cdot \prod_{j \neq i} (\alpha_i - \alpha_j) \right)^{-1}$.

Before stating the main theorem of this section, we require a definition. Let $\phi : R \to S$ be the generalized Frobenius automorphism, which is given by

$$a_0 + a_1 p + \cdots + a_{k-1} p^{k-1} \overset{\phi}{\mapsto} a_0^q + a_1^q p + \cdots + a_{k-1}^q p^{k-1},$$

where $q = p^{\ell}$ is the cardinality of the residue field of S, and each a_i belongs to $\{0, 1, \xi, \dots, \xi^{p^{m \cdot \ell} - 2}\}$, with $\xi \in R^*$ a non-zero element of order $p^{m \cdot \ell} - 1$, as guaranteed by Theorem 2. This allows us to define the generalized trace function $\mathrm{Tr} : R \to S$ of the Galois ring extension R/S, which is given by $\mathrm{Tr}(x) = x + \phi(x) + \cdots + \phi^{m-1}(x)$, where ϕ^i stands for ϕ applied i times. It is easy to see that the generalized trace is a S-linear surjective map [19].

With these definitions and the lemma above at hand, we are ready to prove that generalized Reed-Solomon codes over *Galois Rings* have linear repair, as stated in the theorem below.

Theorem 5. *Let $\alpha_0, \alpha_1, \dots, \alpha_n \in R$ be defined as $\alpha_0 = 0$ and $\alpha_i = \xi^{i-1}$ for $i = 1, \dots, n$. This constitutes an exceptional sequence. Let $\mathbf{y} = (y_0, \dots, y_n) \in (R^*)^{n+1}$ be a vector of units, and let $t \geq 0$ be an integer with $q^{m-1} \leq n - t$. Then the generalized Reed-Solomon code*

$$C = \left\{ (y_0 f(0), y_1 f(\alpha_1), \dots, y_n f(\alpha_n)) \;\middle|\; f \in R[X]_{\leq t} \right\}$$

over R of length $n+1$ and rank $t+1$, has linear repair over S of the 0-th coordinate.

Proof. For $g \in R$ define

$$h_g(X) = g + \phi(g)X^{q-1} + \cdots + \phi^{m-1}(g)X^{q^{m-1}-1}.$$

Observe that $h_g(0) = g$ and $\deg h_g(X) < q^{m-1}$. Now, let D_0 be the S-linear code defined as

$$D_0 = \{(w_0 h_g(\alpha_0), \dots, w_n h_g(\alpha_n)) : g \in R\},$$

where $w_i = \left(y_i \cdot \prod_{j \neq i}(\alpha_i - \alpha_j)\right)^{-1}$. We claim that D_0 satisfies the conditions of Theorem 4. This, from the same theorem, would prove that C has linear repair over S of the 0-th coordinate.

For the first property we need to show that $\pi_0(D_0) = R$, which follows from the fact that $\pi_0(D_0) = w_0 \cdot R \cong R$, since w_0 is a unit. It only remains to show that, for $i = 1, \dots, n$, $\pi_i(D_0) \cong p^j S$ for some $0 \leq j \leq k$. To see this, first observe that, for every $g \in R$, it holds that $\mathrm{Tr}(g \cdot \xi^j) = \xi^j \cdot h_g(\xi^j)$, which follows from the fact that $\phi^u(g \cdot \xi^j) = \phi^u(g) \cdot (\xi^j)^{q^u}$. From this it follows that $\alpha_i \cdot h_g(\alpha_i) = \mathrm{Tr}(g \cdot \alpha_i)$, or $h_g(\alpha_i) = \mathrm{Tr}(g \cdot \alpha_i)/\alpha_i$, since $\alpha_i \in R^*$. From this, and from the fact that the generalized trace function is surjective, we see that $\pi_i(D_0) = \{w_i h_g(\alpha_i) : g \in R\} = \frac{w_i}{\alpha_i} S \cong S$, since w_i and α_i are both units.

Finally, from Lemma 3, and given that $q^{m-1} \leq n - t$, we see that D_0 is an S-submodule of $C^{\perp}$. This concludes the proof. ◻

Remark 1. If $k = 1$, that is, if R is a field, then the Theorem above would hold even if $\alpha_0, \dots, \alpha_n \in R$ is *any* exceptional sequence with $\alpha_0 = 0$. For general Galois rings, the requirement that $\alpha_i \in \{1, \xi, \dots, \xi^{p^d-2}\}$ is crucial for the result to hold. This is because, in general, the relation $\alpha_i \cdot h_g(\alpha_i) = \mathrm{Tr}(g \cdot \alpha_i)$ is not true.

4 Protocols

Let $P_1, \dots, P_n$ be parties connected by secure pairwise communication channels, as well as a broadcast channel. We develop a protocol that allows the parties to obliviously evaluate an arbitrary arithmetic circuit over $\mathbb{Z}/p^k\mathbb{Z}$ on many vectors of inputs in parallel in $d + O(1)$ rounds, where d is the depth of the circuit. Security is defined against a computationally unbounded adversary that can statically corrupt a minority $t < n/2$ of parties and obtain full control, as well as force the computation to abort.

Let m be a positive integer such that $p^{m-1} \leq n - t$; asymptotically we can find $m = \Omega(\log(n))$. In this section we write $S := \mathbb{Z}/p^k\mathbb{Z}$ and let $R = \mathrm{GR}(p^k, m)$ be the degree-m extension ring of S. Let $[\cdot]$ denote the secret-sharing scheme associated to the rank-$(t+1)$ length-$(n+1)$ Reed-Solomon code over R from Theorem 5 with $y_1 = \cdots = y_n = 1$ and some coordinates $\alpha_1, \dots, \alpha_n \in R$ as in Theorem 5. More precisely, for a secret $x \in R$ we denote by $[x]$ a vector of shares $(x_1, \dots, x_n) \in R^n$ such that there is a polynomial $f(X) \in R[X]_{\leq t}$ with $x_i = f(\alpha_i)$ for all i and $f(0) = x$. Whenever we discuss secret-sharings $[x]$, we implicitly mean that each party P_i has the share x_i.

Let $w_0, w_1, \ldots, w_n \in R^*$ be the weights associated to the dual of the Reed-Solomon code above. We can glean explicit compression functions from the proofs of Theorems 4 and 5. Concretely, we set $\phi_i(x_i) = \mathrm{Tr}\left(\frac{w_i x_i}{\alpha_i}\right)$ and $z_i = -\frac{\alpha_i}{w_0}$ for each i. Then, for all share vectors $[x] = (x_1, \ldots, x_n) \in C$, we can reconstruct the secret from the compressed shares as

$$x = \sum_{i=1}^{n} \phi_i(x_i) z_i = -\sum_{i=1}^{n} \mathrm{Tr}\left(\frac{w_i x_i}{\alpha_i}\right) \frac{\alpha_i}{w_0}.$$

4.1 Single-Round Opening and *R*-Multiplication

The repair property of C allows us to efficiently open secret-shared values.

Protocol 1. Open $[x]$ using compressed shares.
Input: $[x] = (x_1, \ldots, x_n)$.

---

1. Each party P_j sends its compressed share $\phi_j(x_j)$ to all other parties.
2. Each party calculates $x = \sum_{j=1}^{n} \phi_j(x_j) z_j$, or aborts if any of the shares are missing or malformed.

Protocol 1 communicates $O(n^2 \log |S|)$ bits in one round, which represents an improvement over the current best known (naive) $O(n^2 \log |R|)$ bits.

Note that the compressed shares do not offer error detection. As such, any maliciously corrupted party P_j may send a value different from $\phi_j(x_j)$, which causes a different value $x' \neq x$ to be opened. Moreover, P_j may send a different value to each party and cause different honest parties to compute different values for x'.

To check whether parties have behaved correctly in Protocol 1, we present Protocol 2. In this separate protocol, we are able to batch check many openings at once at constant cost, so this does not affect our per-gate communication.

For the protocol, we assume access to a functionality $\mathcal{F}_{\mathsf{coin}}$ that samples a uniformly random value in R and sends this value to all parties.

Protocol 2. Check whether sharings $\{[x_\ell]\}_{\ell=1}^N$ where opened correctly. Input: sharings $\{[x_\ell]\}_{\ell=1}^N$ and the values $\{x'_\ell\}_{\ell=1}^N$ they were opened to. Note each party may input a different value x'_ℓ.

Broadcast check
Let $m_\ell^{(i)} \in S$ denote the correct compressed share that P_i was supposed to send during the opening of x_ℓ, and let $\hat{m}_\ell^{i\to j} \in S$ denote the value that was actually sent to P_j.

1. The parties perform N calls to $\mathcal{F}_{\mathsf{coin}}$ to get $s_1, \ldots, s_N \in R$.
2. Each party P_i broadcasts $\gamma_i = \sum_{\ell=1}^N s_\ell \cdot m_\ell^{(i)}$.
3. If some P_j detects that $\gamma_i \neq \sum_{\ell=1}^N s_\ell \cdot \hat{m}_\ell^{i\to j}$, then it aborts.

Consistency check
1. The parties perform N calls to $\mathcal{F}_{\mathsf{coin}}$ to get $r_1, \ldots, r_N$.
2. The parties compute $[v] := \sum_{\ell=1}^N r_\ell([x_\ell] - x'_\ell)$. After the broadcast check has passed, the values x'_ℓ are the same for each party.
3. Each party broadcasts their (uncompressed) share of $[v]$.
4. Each party checks whether the received shares of $[v]$ form a correct sharing of 0. If it does not, they abort.

Remark 2. The number of calls to $\mathcal{F}_{\mathsf{coin}}$ in Protocol 2 can be reduced by the following techniques:

- One can re-use the $s_1, \ldots, s_N$ from the broadcast check in the consistency check, that is, the parties can set $r_\ell = s_\ell$ for $\ell = 1, \ldots, N$.
- Instead of using $s_1, \ldots, s_\ell$ as independent random values, the parties can set $s_\ell := s^{\ell-1}$ for one single random value s. This is at the expense of increasing the cheating probability of the adversary by a polynomial factor of N.

We now show that Protocol 2 is statistically secure, and the probability that an adversary successfully cheats is at most $1/p^m$. To get negligible error probability, in practice we can interpret our secret-sharings over, and move $\mathcal{F}_{\mathsf{coin}}$ to, a Galois ring extension K/R of degree d such that dm is larger than the security parameter κ. If N is large, we can even pack d elements of R into K so this can be done at no extra cost [1].

Proposition 1. *After an execution Protocol 2 where no party aborts, we have $x'_\ell = x_\ell$ for all $\ell = 1, \ldots, N$, except with probability at most $1/p^m$.*

Proof. We first show that if no party aborts the broadcast check, then for each corrupt party P_i and each pair of honest parties $P_j, P_{j'}$, we have that $\hat{m}_\ell^{i\to j} = \hat{m}_\ell^{i\to j'}$ for all $\ell = 1, \ldots, N$. We argue by contradiction: assume we have i, j, j', ℓ^* such that $\hat{m}_{\ell^*}^{i\to j} \neq \hat{m}_{\ell^*}^{i\to j'}$. Since P_i is corrupt, it may have introduced a non-zero error ε_i and broadcast $\gamma_i + \varepsilon_i$ instead of γ_i. Since neither P_j nor $P_{j'}$

aborted, we have that

$$\sum_{\ell=1}^{N} s_\ell \cdot \hat{m}_\ell^{i\to j} = \gamma_i + \varepsilon_i = \sum_{\ell=1}^{N} s_\ell \cdot \hat{m}_\ell^{i\to j'}.$$

This implies that $\sum_{\ell=1}^{N} s_\ell \cdot (\hat{m}_\ell^{i\to j'} - \hat{m}_\ell^{i\to j}) = 0$, and since $s_{\ell^*} \in R$ is uniformly random and independent of the values sent during the opening, this is satisfied with probability at most p^{-m} [1].

Assume the broadcast check passed, and so each honest party P_j received the same value x'_ℓ for each ℓ. If the consistency check also passed without an abort, we know that in the reconstruction of $[v] := \sum_{\ell=1}^{N} r_\ell([x_\ell] - x'_\ell)$, the opened value is exactly equal to v due to the error-correction properties of Shamir secret-sharing. This implies that $\sum_{\ell=1}^{N} r_\ell(x_\ell - x'_\ell) = 0$, which by similar reasoning to the above implies $x_\ell = x'_\ell$ for all $\ell = 1, \ldots, N$ except with probability at most p^{-m}. □

With Protocol 1 we can instantiate the secure multiplication of elements in R in one single round. For example, we can use Beaver multiplication triples, which are sharings $([a], [b], [c])$, with a, b independent and uniformly random and $c = ab$. To securely multiply two sharings $[x], [y]$ using a multiplication triple, the parties open (in one round) $u = [x] - [a]$ and $v = [y] - [b]$ and calculate $[xy] = uv + v[a] + u[b] + [c]$. The protocol we will describe below in Section 4.2 is a bit more involved than what we sketched above, given that it is our goal to postpone the use of the broadcast channel until all multiplications have been performed, and this turns out to lead to selective failure attacks in which an adversary can learn sensitive information if one uses Beaver triples directly.

Finally, since we insist on a maximal adversary, we cannot use random double sharings instead of multiplication triples. Random double sharings are uniformly random sharings $[r]$, together with a "product sharing" of r, i.e. a share vector in the square code C^2. They have the advantage of allowing for a simpler multiplication protocol, that also extends to the inner product of two secret-shared vectors at no extra cost. The square code C^2 is also Reed-Solomon and therefore could have linear repair. However, the rank of the square code is $2t + 1$, and for the maximal adversary $n = 2t + 1$ we cannot have $p^{m-1} \leq n - 2t = 1$.[11]

4.2 Parallel Multiplications

Let $\phi : S^\ell \to R, \psi : R \to S^\ell$ be an (ℓ, m)-RMFE for the Galois ring extension R/S, with $\ell = O(m)$. We can embed two ℓ-length vectors $\mathbf{x}, \mathbf{y} \in S^\ell$ using ϕ and use multiplication of elements in R to obtain the coordinatewise product $\mathbf{x} * \mathbf{y} = \psi(\phi(\mathbf{x})\phi(\mathbf{y}))$. By secret-sharing $[\phi(\mathbf{x})], [\phi(\mathbf{y})]$ we can therefore use secure multiplication in R as explained above, and then open the result and apply ψ to securely evaluate $\mathbf{x} * \mathbf{y}$.

[11] Also note that product sharings do not give error detection, so if we did not insist on a maximal adversary and wanted to use random double sharings, we would have to employ different techniques to get active security.

Unfortunately, this only works for a single multiplication, since in general $\mathbf{x} * \mathbf{y} * \mathbf{z} \neq \psi(\phi(\mathbf{x})\phi(\mathbf{y})\phi(\mathbf{z}))$. The problem is that $\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) \in R$ is not generally contained in the image of ϕ. We get around this problem by "re-encoding" values, as follows.[12]

We encode each input $\mathbf{x} \in S^\ell$ of the circuit by an element in $\psi^{-1}(\mathbf{x}) \in R$. Surjectivity of ψ follows from the definitions of an RMFE hence such an element always exists, though it need not be unique. We proceed to apply the S-linear map $\tau := \phi \circ \psi$, and then multiply the resulting values in R. This allows us to maintain the following invariant: the value on each wire is a vector $\mathbf{x} \in S^\ell$, encoded as $x \in R$ such that $\psi(x) = \mathbf{x}$. This is because if $\mathbf{x}, \mathbf{y} \in S^\ell$ with $\psi(\mathbf{x}) = x$ and $\psi(\mathbf{y}) = y$, then

$$\psi\left(\tau(x)\tau(y)\right) = \psi\left(\phi(\mathbf{x})\phi(\mathbf{y})\right) = \mathbf{x} * \mathbf{y}.$$

Following this invariant, we can define an S-linear secret-sharing scheme of vectors $\mathbf{x} \in S^\ell$, with the share vector given by $[x] \in C$ such that $\psi(x) = \mathbf{x}$. Note that since ϕ, ψ are not in general R-linear (but instead S-linear), parties cannot apply τ to secret-shared values without any interaction.

Our multiplication protocol uses an input-independent offline phase, where secret-shared random elements are generated that are used in the online phase once the inputs are known. The offline phase generates quintuples

$$([a], [b], [\tau(a)], [\tau(b)], [\tau(a)\tau(b)]),$$

where $a, b \in R$ are independent and uniformly random. Because all quintuples can be generated in parallel, the round complexity is not important, and so we can use known techniques. The only requirement is that the generated quintuples are correct; for example, we can use from [1] the protocol `RandElStat` to get pairs $([a], [\tau(a)])$, and combine it with their multiplication protocol to generate the quintuples.

Given a quintuple, the online single-round protocol to compute $[\tau(x)\tau(y)]$ is very similar to using a regular multiplication triple. The parties open $u = [x] - [a]$ and $v = [y] - [b]$ using Protocol 1 and then compute

$$[\tau(x)\tau(y)] = \tau(v)[\tau(a)] + \tau(u)[\tau(b)] + [\tau(a)\tau(b)] + \tau(u)\tau(v).$$

This operation is definitely secure against a passive adversary, however against an active adversary the security is not so clear. Since we postpone the broadcast and consistency checks until the end, an active adversary can introduce additive errors when opening $[u], [v]$, and it can do so at every multiplication throughout the circuit. The question whether this compromises the privacy or not is closely related to the amount of redundancy the compressed shares contain. We leave this question for now, and return to it in Section 5.

[12] The idea of re-encoding is based on [4], but we have developed an improved encoding that allows us to decrease the number of rounds for a multiplication, and that also allows for a simpler input phase. We explain the differences between the two approaches in Section 5.

Our protocol circumvents this issue entirely, by observing that the issue does not arise when using *function-dependent preprocessing* [2]. Whereas Beaver multiplication triples are generic in the sense that the triples are not specific to any particular multiplication gate, with function-dependent preprocessing we generate random sharings tailored to each multiplication gate. Besides sidestepping the issue of security, this also has the advantage that it reduces communication by half, and it allows computing the inner product of two secret-shared vectors at the same communication cost as one secure multiplication.

Recall each wire in our circuit has an associated secret vector $\mathbf{x} \in S^\ell$. We will maintain the invariant that each such vector $\mathbf{x}$ is secret-shared as the tuple $[\![\mathbf{x}]\!] := ([\lambda_x], \mu_x)$, where $\lambda_x \in R$ is a uniformly random element that is secret-shared using the scheme $[\cdot]$, and $\mu_x = x - \lambda_x$ is a publicly known element such that $\psi(x) = \mathbf{x}$. Given λ_x and μ_x, a party can compute $\mathbf{x} = \psi(\mu_x + \lambda_x)$ and recover the secret value. This construction defines an S-linear secret-sharing scheme.

To securely compute $[\![\mathbf{x} * \mathbf{y}]\!]$ from $[\![\mathbf{x}]\!]$ and $[\![\mathbf{y}]\!]$, the parties can proceed as in Protocol 3. We assume a functionality $\mathcal{F}_{\mathsf{prep}}$ that generates the necessary preprocessing material, consisting of the following:

- A shared random value $[\lambda_x]$ for every wire.
- For every multiplication gate with inputs x and y, a quintuple

$$([\lambda_x], [\lambda_y], [\tau(\lambda_x)], [\tau(\lambda_y)], [\tau(\lambda_x)\tau(\lambda_y)]).$$

Generating these sharings can be done in a similar way as the quintuples mentioned above. Note that this data is specific to the topology of the circuit.

Protocol 3. Multiply $[\![\mathbf{x}]\!] = ([\lambda_x], \mu_x)$ and $[\![\mathbf{y}]\!] = ([\lambda_y], \mu_y)$.
Preprocessed: $[\lambda_z]$ and $([\lambda_x], [\lambda_y], [\tau(\lambda_x)], [\tau(\lambda_y)], [\tau(\lambda_x)\tau(\lambda_y)])$ produced by $\mathcal{F}_{\mathsf{prep}}$.

---

1. Use Protocol 1 to open μ_w towards all parties, where
$\mu_w = \tau(\mu_x)\tau(\mu_y) + \tau(\mu_x)[\tau(\lambda_y)] + \tau(\mu_y)[\tau(\lambda_x)] + [\tau(\lambda_x)\tau(\lambda_y)] - [\lambda_w]$.
2. The parties return the shares $[\![\mathbf{w}]\!] := ([\lambda_w], \mu_w)$.

The reason why this multiplication protocol is private against an active adversary, is the following. The adversary can still tamper with the reconstruction of μ_w by adding some error (perhaps different to every party). However, here this error is independent of any sensitive input and is in fact completely known by the adversary, so intuitively speaking, it does not allow the adversary to learn anything new. Formally speaking, the simulator will be able to extract these errors and emulate the honest parties correctly in the ideal world.

4.3 Secure Parallel Computation

Let f be a function represented by an arithmetic circuit over S. For simplicity and without loss of generality, we assume f is of the form $S^n \to S$, where each input in S is held by one party. We show how to securely evaluate f in parallel ℓ times in Protocol 4 below. Here, let $\mathbf{x}_i \in S^\ell$ be the vector of inputs of party P_i for the ℓ executions of f. Also, let $\mathbf{w}_j \in S^n$ denote the input vector to the j-th execution of f (that is, the i-th entry of $\mathbf{w}_j$ is the j-th entry of $\mathbf{x}_i$).

Our protocol, described below, requires as preprocessing material a random sharing $[r]$ where $r \in \ker(\psi)$, which is used to mask the output before opening. This is required since, due to the invariant, the value that is produced as output is a preimage under ψ of the output vector, but this preimage itself may depend on intermediate values. By masking before opening with an element of the kernel of ψ, we ensure that this preimage is uniform among all possible preimages, which enables the simulator in the proof to emulate this step towards the adversary.

Protocol 4. Obliviously evaluate f on ℓ vectors of inputs $\mathbf{w}_1, \ldots, \mathbf{w}_\ell \in S^n$.
Preprocessing: Random sharing $[r]$, where $r \in \ker(\psi)$.
Output: $f(\mathbf{w}_1), \ldots, f(\mathbf{w}_\ell)$.

Input phase. Let $\mathbf{x}_i \in S^\ell$ be the input from P_i. For each $i = 1, \ldots, n$, the parties do the following.
1. The parties send their shares of $[\lambda_{x_i}]$ to P_i.
2. P_i broadcasts $\mu_{x_i} := x_i - \lambda_{x_i}$, where $\psi(x_i) = \mathbf{x}_i$.
3. Parties set $[\![\mathbf{x_i}]\!] = ([\lambda_{x_i}], \mu_{x_i})$.

Computation phase. The parties process the circuit gate-by-gate, using Protocol 3 and maintaining the invariant of $[\![\cdot]\!]$.

Output phase.
1. The parties run Protocol 2 to verify correctness of all opened values.
2. Let $[\![\mathbf{z}]\!] = ([\lambda_z], \mu_z)$ be the shared output. Each party broadcasts their shares of $[\lambda_z] + [r]$, and the output is defined as $\mathbf{z} = \psi((\lambda_z + r) + \mu_z)$.

The security of our protocol is proved in the following theorem, which in turn proves Theorem 1.

Theorem 6. *Protocol 4 securely computes ℓ copies of the function f in the $(\mathcal{F}_{\mathsf{coin}}, \mathcal{F}_{\mathsf{prep}})$-hybrid model with statistical security with abort against an active adversary.*

Proof. We construct a simulator $\mathcal{S}$ that interacts with an environment $\mathcal{Z}$ and with an MPC functionality in such a way that the environment cannot distinguish between the simulated execution and the real protocol.

The simulator emulates the behavior of the honest parties towards the adversary, as well as emulating the functionalities $\mathcal{F}_{\mathsf{coin}}$ and $\mathcal{F}_{\mathsf{prep}}$. $\mathcal{S}$ begins by generating all the necessary preprocessing material. For the input phase, the simulator receives $\mu_{x_i} := x_i - \lambda_{x_i}$ for each corrupt party P_i, and since $\mathcal{S}$ knows λ_{x_i}, it can recover the inputs $\mathbf{x}_i$, which are then sent to the MPC functionality. For the inputs from the honest parties $\mathcal{S}$ simply uses dummy values.

For the addition gates the simulator simply emulates the local operations on the honest parties. On the other hand, for multiplication gates, $\mathcal{S}$ receives the reduced shares $\phi_i(\mu_{w,i})$ from each corrupt party P_i, corresponding to the secrets $\mu_w = x - \lambda_x$ from the protocol. Here the simulator samples uniformly at random some value $\mu_w \in R$, and sets the honest parties' shares so that they are consistent with this value and with the shares held by the corrupt parties. Then $\mathcal{S}$ opens these (compressed) shares towards the adversary.

Observe that the adversary may cause each corrupt party P_i to send $\phi_i(\mu_{w,i})+\epsilon_{ij}$ to each honest party P_j, instead of the correct reduced share. As a result, P_j will think that μ_w is actually equal to $\sum_{\ell=1}^{n}(\phi_\ell(\mu_{w,\ell})+\epsilon_{\ell j})\cdot z_\ell$. Since the simulator knows the actual shares that the corrupt parties must have sent, it knows the value described above and it can continue emulating the honest parties.[13]

For the output phase the simulator begins by sending an abort signal if there exists a corrupt party P_i and a pair of different honest parties P_j and $P_{j'}$ such that $\epsilon_{ij} \neq \epsilon_{ij'}$, for some multiplication gate. If this is not the case let us denote $\epsilon_i := \epsilon_{ij}$. If $e := \sum_{\ell=1}^{n} \epsilon_\ell \cdot z_\ell \neq 0$, then $\mathcal{S}$ sends an abort signal. Else, $\mathcal{S}$ receives the output from the MPC functionality, samples a random preimage under ψ of this output, sets the shares of the honest parties so that they are consistent with this output and with the corrupt parties' shares, and then emulates the reconstruction protocol by broadcasting the shares corresponding to the honest parties.

Now we argue that the simulation is statistically indistinguishable to the environment from the real execution. The input phase is clearly indistinguishable, as well as the additions as they follow the exact same distribution in both executions. For multiplications, observe that in the real world the adversary only sees $\mu_w = w - \lambda_w$, but since λ_w is uniformly random and unknown to the adversary, then μ_w follows the uniform distribution, which coincides with what the adversary sees in the ideal execution.

The only potential difference lies in the check performed by the parties at the end. In the ideal execution the parties abort if any of the broadcasted values does not match, which is also the case in the real execution except with negligible probability thanks to Proposition 1 combined with the remark about moving to a large Galois ring. $\square$

5 Discussion

Our results demonstrate that regenerating codes can improve the round complexity of MPC protocols, or alternatively, when insisting on a minimum number of rounds they can improve the communication complexity.

Differences with [4]. Our protocol uses techniques from [4] and the generalizations to Galois rings [1, 8], but there are a few key differences. Evidently, we have

[13] This is precisely what goes wrong if one uses traditional multiplication triples: The error on each honest party's share will depend on the honest parties' inputs, which the adversary cannot simulate.

plugged in Protocol 1 to get an efficient single-round opening. But the essential modification is that we encode vectors in S^ℓ not using ϕ, but using ψ^{-1}. The difference is subtle, but it allows us to combine a multiplication together with `ReEncode` procedure from [4], that applies the map $\tau = \phi \circ \psi$ to the output of a multiplication, into a single round. The original work also benefits from this approach since it improves the number of rounds, even without using regenerating codes. Additionally, this encoding simplifies correctness of the input phase, since the security of the protocol does not require each wire value to be contained in the image of ϕ.

Active security of the quintuple-based protocol. The quintuple-based protocol sketched in Section 4.2 is passively secure, but it is not clear whether it is private against a malicious adversary that introduces errors. Recall that to multiply $[x]$ and $[y]$, the parties open $u = [x] - [a]$ and $v = [y] - [b]$ using Protocol 1 and then compute $[\tau(x)\tau(y)] = \tau(v)[\tau(a)] + \tau(u)[\tau(b)] + [\tau(a)\tau(b)] + \tau(u)\tau(v)$. Assume without loss of generality that P_1 is honest. Since the adversary may send different compressed shares to different parties, it may cause P_1 to think that u and v are actually equal to $u + \varepsilon$ and $v + \delta$, which causes P_1's share to be equal to the correct share plus

$$e_1 = \tau(\varepsilon)\tau(b)_1 + \tau(\delta)\tau(a)_1 + \tau(u)\tau(\delta) + \tau(v)\tau(\varepsilon) + \tau(\varepsilon)\tau(\delta).$$

Assume the other honest parties learn u and v correctly. Notice that the adversary does not know the value of e_1 as it depends on the unknown values a and b.

Now imagine that as part of the function being computed, $w = \tau(x)\tau(y)$ is fed into another multiplication gate. As part of the protocol, the parties open $u' = [w] - [a']$. Assume for a moment that instead of sending compressed shares, the parties send their full shares. Hence, the adversary receives the share of $w - a'$ from P_1, which is altered by an amount of e_1. However, note that the adversary knows the corrupt parties' shares of $w - a'$, so if all the shares sent by the honest parties happen to be consistent with these shares, then the adversary can conclude that $e_1 = 0$, else, $e_1 \neq 0$ (and in this case the parties abort).

Notice that the adversary knows all values in the expression defining e_1 except for $\tau(a)_1$ and $\tau(b)_1$. Consider for simplicity that $\delta = 0$, so $e_1 = \tau(\varepsilon)\tau(b)_1 + \tau(v)\tau(\varepsilon)$. If ε is such that $\tau(\varepsilon) \in R^*$, knowing whether e_1 is non-zero or not leaks one bit of information about $\tau(b)_1$, namely whether it equals $-\tau(v)$ or not. However, to the adversary, $\tau(a)$ is a function of $\tau(a)_1$, so this leaks information about $\tau(a)$, which in turn leaks information about $\tau(y)$ given that the adversary knows $\tau(v)$ and $v = y - b$.

The attack above assumes that the parties send their full shares, but in the protocol description they send their *reduced* shares. One may think that these reduced shares, since they carry less redundancy than the original shares, may hinder the attack. However, it seems hard to quantify this, since for example a repair scheme with no compression (so ϕ_i is the identity for all i) would be susceptible to the attack above, and it is not clear at what point the compression level is "enough" so that the adversary cannot learn any sensitive information. We leave this for future work.

Lower bound. We have shown that when amortizing over many parallel multiplications, we can multiply two elements of $\mathbb{Z}/p^k\mathbb{Z}$ in a single round with $o(n^2)$ elements of $\mathbb{Z}/p^k\mathbb{Z}$ communication, in our model of honest majority and unconditional security. To the best of our knowledge, there is no currently known way to achieve $o(n^2)$ complexity without amortization, and we conjecture that this is in fact impossible.

To support this conjecture, we note that a single-round opening of a sharing over any *fixed* Galois ring requires $\Omega(n^2)$ bits of communication. This is because each party must hear from at least t other parties; if not, an adversary could corrupt t parties and learn the secret without opening the value. All multiplication protocols in the preprocessing model rely on opening a value, and the alternative of re-sharing also requires each party to send $\Omega(n)$ shares to the other parties.

Limitation on the possible values of n and t. Notice that, once the values for p and k have been set, Theorem 5 implicitly contains certain restrictions on the possible values of t and n. First, we have the bound $p^{\ell\cdot(m-1)} \leq n - t$, but in addition, since we assume that $\alpha_0, \dots, \alpha_n$ constitutes an exceptional sequence, it must be the case that $n + 1 \leq p^{m\cdot\ell}$. By considering a maximal adversary $n = 2t + 1$, we get that $p^{\ell\cdot(m-1)} - 1 \leq t \leq \frac{p^{\ell\cdot m}-2}{2}$.

It holds that $\ell = 1$ if $S = \mathbb{Z}/p^k\mathbb{Z}$, but if a larger value of n (or equivalently, t) must be accommodated, ℓ can be chosen to be large enough so that the bounds above are satisfied. This would lead to an equivalent asymptotic result as the one obtained here for secure computation over $\mathrm{GR}(p^k, \ell)$, which can be translated without asymptotic loss into a result for (parallel) computation over $\mathbb{Z}/p^k\mathbb{Z} = \mathrm{GR}(p^k, 1)$ by using, once again, RMFEs, exactly as done in [8].

6 Acknowledgments

We thank the anonymous Asiacrypt 2021 reviewers for their valuable feedback. Chaoping Xing's research work is partially supported by the NSFC under grant 12031011, Huawei-SJTU joint projects and the National Key Research and Development Project 2020YFA0712300. During his time in Aarhus University, Daniel Escudero was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 669255 (MPCPRO).

This paper was prepared for information purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co and its affiliates ("JP Morgan"), and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under

References

[1] Mark Abspoel, Ronald Cramer, Ivan Damgård, Daniel Escudero, and Chen Yuan. “Efficient Information-Theoretic Secure Multiparty Computation over $\mathbb{Z}/p^k\mathbb{Z}$ via Galois Rings”. In: *TCC 2019, Part I*. Ed. by Dennis Hofheinz and Alon Rosen. Vol. 11891. LNCS. Springer, Heidelberg, Dec. 2019, pp. 471–501. DOI: 10.1007/978-3-030-36030-6_19.

[2] Aner Ben-Efraim, Michael Nielsen, and Eran Omri. “Turbospeedz: Double Your Online SPDZ! Improving SPDZ Using Function Dependent Preprocessing”. In: *ACNS 19*. Ed. by Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung. Vol. 11464. LNCS. Springer, Heidelberg, June 2019, pp. 530–549. DOI: 10.1007/978-3-030-21568-2_26.

[3] Fabrice Benhamouda, Akshay Degwekar, Yuval Ishai, and Tal Rabin. “On the local leakage resilience of linear secret sharing schemes”. In: *Journal of Cryptology* 34.2 (2021), pp. 1–65.

[4] Ignacio Cascudo, Ronald Cramer, Chaoping Xing, and Chen Yuan. “Amortized Complexity of Information-Theoretically Secure MPC Revisited”. In: *CRYPTO 2018, Part III*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10993. LNCS. Springer, Heidelberg, Aug. 2018, pp. 395–426. DOI: 10.1007/978-3-319-96878-0_14.

[5] Hervé Chabanne, Houssem Maghrebi, and Emmanuel Prouff. “Linear repairing codes and side-channel attacks”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2018), pp. 118–141.

[6] Arka Rai Choudhuri, Aarushi Goel, Matthew Green, Abhishek Jain, and Gabriel Kaptchuk. *Fluid MPC: Secure Multiparty Computation with Dynamic Participants*. Cryptology ePrint Archive, Report 2020/754. https://eprint.iacr.org/2020/754. 2020.

[7] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015. ISBN: 9781107043053. URL: http://www.cambridge.org/de/academic/subjects/computer-science/cryptography-cryptology-and-coding/secure-multiparty-computation-and-secret-sharing?format=HB&isbn=9781107043053.

[8] Ronald Cramer, Matthieu Rambaud, and Chaoping Xing. *Asymptotically-Good Arithmetic Secret Sharing over $Z/(p^\ell Z)$ with Strong Multiplication and Its Applications to Efficient MPC*. Cryptology ePrint Archive, Report 2019/832. https://eprint.iacr.org/2019/832. 2019.

[9] Ivan Damgård, Kasper Green Larsen, and Jesper Buus Nielsen. “Communication Lower Bounds for Statistically Secure MPC, With or Without Preprocessing”. In: *CRYPTO 2019, Part II*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11693. LNCS. Springer, Heidelberg, Aug. 2019, pp. 61–84. DOI: 10.1007/978-3-030-26951-7_3.

[10] Ivan Damgård and Jesper Buus Nielsen. “Scalable and unconditionally secure multiparty computation”. In: *Annual International Cryptology Conference*. Springer. 2007, pp. 572–590.

[11] Daniel Escudero and Anders Dalskov. *Honest Majority MPC with Abort with Minimal Online Communication*. Cryptology ePrint Archive, Report 2020/1556. `https://eprint.iacr.org/2020/1556`. 2020.

[12] Venkatesan Guruswami and Mary Wootters. “Repairing Reed-Solomon Codes”. In: *CoRR* abs/1509.04764 (2015). Note, we specifically refer to the version published on arXiv. arXiv: `1509.04764`.

[13] Venkatesan Guruswami and Mary Wootters. “Repairing Reed-solomon codes”. In: *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*. Ed. by Daniel Wichs and Yishay Mansour. ACM, 2016, pp. 216–226. DOI: `10.1145/2897518.2897525`. URL: `https://doi.org/10.1145/2897518.2897525`.

[14] Yuval Ishai and Eyal Kushilevitz. “On the hardness of information-theoretic multiparty computation”. In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2004, pp. 439–455.

[15] Guillaume Quintin, Morgan Barbier, and Christophe Chabot. “On Generalized Reed–Solomon Codes Over Commutative and Noncommutative Rings”. In: *IEEE transactions on information theory* 59.9 (2013), pp. 5882–5897.

[16] Korlakai Vinayak Rashmi, Nihar B Shah, and P Vijay Kumar. “Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction”. In: *IEEE Transactions on Information Theory* 57.8 (2011), pp. 5227–5239.

[17] KV Rashmi, Nihar B Shah, Kannan Ramchandran, and P Vijay Kumar. “Regenerating codes for errors and erasures in distributed storage”. In: *2012 IEEE International Symposium on Information Theory Proceedings*. IEEE. 2012, pp. 1202–1206.

[18] Itzhak Tamo, Min Ye, and Alexander Barg. “Optimal repair of Reed-Solomon codes: Achieving the cut-set bound”. In: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2017, pp. 216–227.

[19] Zhe-Xian Wan. *Lectures on Finite Fields and Galois Rings*. World Scientific Publishing Company, 2003. ISBN: 978-9812385048. URL: `https://doi.org/10.1142/5350`.

[20] Avi Wigderson, MB Or, and S Goldwasser. “Completeness theorems for noncryptographic fault-tolerant distributed computations”. In: *Proceedings of the 20th Annual Symposium on the Theory of Computing (STOC’88)*. 1988, pp. 1–10.