Efficient Multi-Party Computation over Rings*

Ronald Cramer[1], Serge Fehr[1], Yuval Ishai[2], and Eyal Kushilevitz[2]

[1] BRICS***, Department of Computer Science, Århus University, Denmark
{cramer,fehr}@brics.dk
[2] Computer Science Department, Technion, Israel
{yuvali,eyalk}@cs.technion.ac.il

Abstract. Secure multi-party computation (MPC) is an active research area, and a wide range of literature can be found nowadays suggesting improvements and generalizations of existing protocols in various directions. However, all current techniques for secure MPC apply to functions that are represented by (boolean or arithmetic) circuits over finite *fields*. We are motivated by two limitations of these techniques:

- Generality. Existing protocols do not apply to computation over more general algebraic structures (except via a brute-force simulation of computation in these structures).
- Efficiency. The best known *constant-round* protocols do not efficiently scale even to the case of large finite fields.

Our contribution goes in these two directions. First, we propose a basis for unconditionally secure MPC over an arbitrary finite *ring*, an algebraic object with a much less nice structure than a field, and obtain efficient MPC protocols requiring only a *black-box access* to the ring operations and to random ring elements. Second, we extend these results to the constant-round setting, and suggest efficiency improvements that are relevant also for the important special case of fields. We demonstrate the usefulness of the above results by presenting a novel application of MPC over (non-field) rings to the round-efficient secure computation of the maximum function.

1 Introduction

Background. The goal of *secure multi-party computation* (MPC), as introduced by Yao [40], is to enable a set of players to compute an arbitrary function f of their private inputs. The computation must guarantee the correctness of the result while preserving the privacy of the players' inputs, even if some of the players are corrupted by an adversary and misbehave in an arbitrary malicious way. Since the initial plausibility results in this area [41, 26, 7, 11], much effort has been put into enhancing these results, and nowadays there is a wide range of literature treating issues like improving the communication complexity (e.g., [24, 25, 28]) or the round complexity (e.g., [1, 5, 3, 30]), and coping with more powerful (e.g., [37, 10, 9]) or more general (e.g., [27, 20, 14]) adversaries.

A common restriction on all these results is that the function f is always assumed to be represented by an arithmetic circuit over a finite *field*, and hence all computations "take place" in this field. Thus, it is natural to ask whether MPC can also be efficiently implemented over a richer class of structures, such as arbitrary finite *rings*. This question makes sense from a theoretical point of view, where it may be viewed as a quest for minimizing the *axioms* on which efficient secure MPC can be based, but also from a practical point of view, since a positive answer would allow greater freedom in the representation of f, which in turn can lead to efficiency improvements. Unfortunately, general rings do not enjoy some of the useful properties of fields on which standard MPC protocols rely: non-zero ring elements may not have inverses (in fact, a ring may even not contain 1, in which case *no* element is invertible), there might exist zero-divisors, and the multiplication may not be commutative. Indeed,

* This is an extended version of [17].

*** Basic Research in Computer Science (www.brics.dk), funded by the Danish National Research Foundation.

already over a relatively "harmless" ring like $\mathbb{Z}_m$, Shamir's secret sharing scheme [39], which serves as the standard building block for MPC, is not secure a-priori. For instance, if m is even and if $p(X)$ is a polynomial over $\mathbb{Z}_m$ hiding a secret s as its free coefficient, then the share $s_2 = p(2)$ is odd if and only if the secret s is odd. Thus, even the most basic tools for secure MPC have to be modified before applying them to the case of rings. A step in this direction was taken in [19, 16], where *additively* homomorphic secret sharing schemes for arbitrary Abelian groups have been proposed. However, this step falls short of providing a complete solution to our problem (which in particular requires both addition and multiplication of shared secrets), and so the question of MPC over rings remains unanswered.

An additional limitation of current MPC techniques which motivates the current work is related to the efficiency of *constant-round* protocols. Without any restriction on the number of rounds, most protocols from the literature generalize smoothly to allow arithmetic computation over arbitrary finite fields. This is particularly useful for the case of "numerical" computations, involving integers or (finite-precision) reals; indeed, such computations can be naturally embedded into fields of a sufficiently large characteristic. However, in the constant-round setting the state of affairs is quite different. All known protocols for efficiently evaluating a circuit in a constant number of rounds [41, 5, 12, 34] are based on Yao's *garbled circuit* construction, which does not efficiently scale to arithmetic circuits over large fields.[1] The only constant-round protocols in the literature which do efficiently scale to arithmetic computation over large fields apply to the weaker computational models of formulas [1] or branching programs [31], and even for these models their complexity is (at least) quadratic in the representation size. Hence, there are no truly satisfactory solutions to the problem of constant-round MPC over general fields, let alone general rings.

Our results. In this paper, we propose a basis for obtaining unconditionally secure MPC over arbitrary finite rings. In particular, we present an efficient MPC protocol that requires only *black-box access* to the ring operations (addition, subtraction, and multiplication) and the ability to sample random ring elements. It is perfectly secure with respect to an active adversary corrupting up to $t < n/3$ of the n players, and its complexity is comparable to the corresponding field-based solutions. This is a two-fold improvement over the classical field-based MPC results. It shows that MPC can be efficiently implemented over a much richer class of structures, namely arbitrary finite rings, and it shows that there exists in fact one "universal" protocol that works for any finite ring (and field). Finally, the tools we provide can be combined with other work on MPC, and hence expand a great body of work on MPC to rings.

On the constant-round front, we make two distinct contributions. First, we show that the feasibility of MPC over black-box rings carries over to the constant-round setting.[2] To this end, we formulate and utilize a *garbled branching program* construction, based on a recent randomization technique from [31]; however, as the algebraic machinery which was originally used in its analysis does not apply to general rings, we provide a combinatorially-oriented presentation and analysis which may be of independent interest. As a second contribution, we suggest better ways for evaluating natural classes of arithmetic formulas and branching programs in a constant number of rounds. In particular, we obtain protocols for *small-width* branching programs and *balanced* formulas in which the communication complexity is nearly linear in their size. The former protocols are based on the garbled branching program construction, and the latter on a combination of a complexity result

[1] It is obviously possible to apply the brute-force approach of simulating each field operation by a boolean circuit computing it. However, this approach is unsatisfactory both from a theoretical point of view (as its complexity grows super-linearly in the length of a field element) and from a practical point of view. The same objection applies to the implementation of *ring* operations using *field* or *boolean* operations. (Due to the lack of provable lower bounds in complexity-theory, one cannot tell for sure whether this is an inherent phenomenon or just a limitation of currently known techniques.)

[2] This is not clear a-priori, and in fact most randomization techniques used in the context of constant-round MPC (e.g.,[1, 21, 3, 30]) clearly do not apply to this more general setting.

from [13] with a variant of randomization technique from [3]. While the main question in this context (namely, that of obtaining efficient constant-round protocols for arithmetic *circuits*) remains open, our techniques may still provide the best available tools for efficiently realizing "numerical" MPC tasks that arise in specific applications. Furthermore, these techniques may also be beneficial in the two-party setting of [38] (via the use of a suitable homomorphic encryption scheme) and in conjunction with computationally-secure MPC (using, e.g., [15]).

We conclude with an example for the potential usefulness of secure MPC over non-field rings. Specifically, we show how to efficiently compute the maximum of n integers with better round complexity than using alternative approaches.

Organization. Section 2 deals with the model. The main body of the paper has two parts corresponding to our two main contributions: the first deals with general MPC over rings (Section 3) and the other concentrates on constant-round protocols (Section 4). Finally, in Section 5 we describe an application of MPC over non-field rings.

2 Model

We consider the *secure-channels model*, as introduced in [7, 11], where a set $\mathcal{P} = \{P_1, \ldots, P_n\}$ of n players is connected by bilateral, synchronous, reliable secure channels. For the case of constant-round secure computation, a broadcast channel is also assumed to be available, while it has to be implemented otherwise. Our goal is to obtain a protocol for securely computing a function given by an arithmetic circuit over an arbitrary ring R. (In Section 4 we will also be interested in functions represented by formulas and branching programs over a ring R.) By default, we consider *unconditional* or *perfect* security against an *adaptive*, *active* adversary. The reader is referred to, e.g., [8] for a definition of secure protocols in this setting. Such a protocol is *black-box* if: (1) its description is independent of R and it only makes black-box calls to the ring operations (addition, subtraction and multiplication) and to random ring elements; and (2) its security holds regardless of the underlying ring R, in the sense that each adversary attacking the protocol admits a simulator having only a black-box access to R.

3 Multi-Party Computation over Rings

3.1 Mathematical Preliminaries

We assume the reader to be familiar with basic concepts of group and ring theory. However, we also make use of the notions of a *module* and of an *algebra*, which we briefly introduce here. Let Λ be a commutative ring with 1. An (additive) Abelian group G is called a Λ-*module* if a *number multiplication* $\Lambda \times G \to G, (\lambda, a) \mapsto \lambda \cdot a$ is given such that $1 \cdot a = a$, $\lambda \cdot (a + b) = (\lambda \cdot a) + (\lambda \cdot b)$, $(\lambda + \mu) \cdot a = (\lambda \cdot a) + (\mu \cdot a)$ and $(\lambda \cdot \mu) \cdot a = \lambda \cdot (\mu \cdot a)$ for all $\lambda, \mu \in \Lambda$ and $a, b \in G$. Hence, loosely speaking, a module is a vector space over a ring (instead of over a field). An arbitrary ring R is called a Λ-*algebra* if (the additive group of) R is a Λ-module and $(\lambda \cdot a) \cdot b = \lambda \cdot (a \cdot b) = a \cdot (\lambda \cdot b)$ holds for all $\lambda \in \Lambda$ and $a, b \in R$.[3] For example, every Abelian group G is a $\mathbb{Z}$-module and every ring R is a $\mathbb{Z}$-algebra; the number multiplication is given by $0 \cdot a = 0$, $\lambda \cdot a = a + \cdots + a$ (λ times) if $\lambda > 0$, and $\lambda \cdot a = -((-\lambda) \cdot a)$ if $\lambda < 0$. We also write λa or $a\lambda$ instead of $\lambda \cdot a$.

[3] Note that even though we have two kinds of addition (addition in Λ and in R) and three kinds of multiplication (multiplication in Λ, in R, and number multiplication), addition is always denoted by "+" and multiplication by "$\cdot$" (or nothing). However, it should always be clear from the context, which addition or multiplication is meant.

3.2 Span Programs over Rings and Linear Secret Sharing

Monotone span programs over (finite) fields were introduced in [32] and turned out to be in a one-to-one correspondence to linear secret sharing schemes (over finite fields). This notion was extended in [16] to monotone span programs over (possibly infinite) *rings*, and it was shown that *integer span programs*, i.e. span programs over $\mathbb{Z}$, have a similar correspondence to *black-box* secret sharing (over arbitrary Abelian groups). We briefly recall some definitions and observations.

Definition 1. *A subset Γ of the power set $2^{\mathcal{P}}$ of $\mathcal{P}$ is called an* access structure *on $\mathcal{P}$ if $\emptyset \notin \Gamma$ and if Γ is closed under taking supersets: $A \in \Gamma$ and $A' \supseteq A$ implies that $A' \in \Gamma$. A subset $\mathcal{A}$ of $2^{\mathcal{P}}$ is called an* adversary structure *on $\mathcal{P}$ if its complement $\mathcal{A}^c = 2^{\mathcal{P}} \setminus \mathcal{A}$ is an access structure.*

Let Λ be an arbitrary (not necessarily finite) commutative ring with 1. Consider a matrix M over Λ with, say, d rows and e columns (this will be denoted as $M \in \Lambda^{d \times e}$), a *labeling function* $\psi : \{1, \dots, d\} \to \mathcal{P}$ and the *target vector* $\boldsymbol{\varepsilon} = (1, 0, \dots, 0)^T \in \Lambda^e$. The function ψ labels each row of M with a number corresponding to one of the players. If $A \subseteq \mathcal{P}$ then M_A denotes the restriction of M to those rows i with $\psi(i) \in A$, and, similarly, if $\mathbf{x}$ denotes an arbitrary d-vector then $\mathbf{x}_A$ denotes the restriction to those coordinates i with $\psi(i) \in A$. In case $A = \{P_i\}$, we write M_i and $\mathbf{x}_i$ instead of M_A and $\mathbf{x}_A$. Finally, $\mathrm{im}(\cdot)$ denotes the image and $\ker(\cdot)$ the kernel (or null-space) of a matrix.

Definition 2. *Let $\mathcal{M} = (\Lambda, M, \psi, \boldsymbol{\varepsilon})$ be a quadruple as above, and let Γ be an access structure on $\mathcal{P}$. Then, $\mathcal{M}$ is called a* (monotone)[4] span program *(over Λ) for the access structure Γ, or, alternatively, for the adversary structure $\mathcal{A} = \Gamma^c$, if for all $A \subseteq \mathcal{P}$ the following holds.*

- *If $A \in \Gamma$, then $\boldsymbol{\varepsilon} \in \mathrm{im}(M_A^T)$, and*
- *if $A \notin \Gamma$, then there exists $\boldsymbol{\kappa} = (\kappa_1, \dots, \kappa_e)^T \in \ker(M_A)$ with $\kappa_1 = 1$.*

In the former case, we say that A is accepted *and in the latter that A is* rejected *by $\mathcal{M}$. If $\Lambda = \mathbb{Z}$ then $\mathcal{M}$ is called an* integer span program, ISP *for short. Finally,* $\mathrm{size}(\mathcal{M})$ *is defined as d, the number of rows of M.*

By basic linear algebra, the existence of $\boldsymbol{\kappa} \in \ker(M_A)$ with $\kappa_1 = 1$ implies that $\boldsymbol{\varepsilon} \notin \mathrm{im}(M_A^T)$, however the other direction generally only holds if Λ is a field.[5]

Let G be an arbitrary finite Abelian group that can be seen as a Λ-module. As a consequence, it is well defined how a matrix over Λ acts on a vector with entries in G. Then, a span program $\mathcal{M} = (\Lambda, M, \psi, \boldsymbol{\varepsilon})$ for an access structure Γ gives rise to a secret sharing scheme for secrets in G: To share $s \in G$, the dealer chooses a random vector $\mathbf{b} = (b_1, \dots, b_e)^T \in G^e$ of group elements with $b_1 = s$, computes $\mathbf{s} = M\mathbf{b}$ and, for every player $P_i \in \mathcal{P}$, hands $\mathbf{s}_i$ (privately) to P_i. This is a secure sharing of s, with respect to the access structure Γ. Namely, if $A \in \Gamma$ then there exists an (A-dependent) vector $\boldsymbol{\lambda}$, with entries in Λ, such that $M_A^T \boldsymbol{\lambda} = \boldsymbol{\varepsilon}$. It follows that s can be reconstructed from $\mathbf{s}_A$ by $\mathbf{s}_A^T \boldsymbol{\lambda} = (M_A \mathbf{b})^T \boldsymbol{\lambda} = \mathbf{b}^T M_A^T \boldsymbol{\lambda} = \mathbf{b}^T \boldsymbol{\varepsilon} = s$. On the other hand, if $A \notin \Gamma$ then there exists an (A-dependent) vector $\boldsymbol{\kappa} \in \Lambda^e$ with $M_A \boldsymbol{\kappa} = \mathbf{0}$ and $\kappa_1 = 1$. For arbitrary $s' \in G$ define $\mathbf{s}' = M(\mathbf{b} + \boldsymbol{\kappa}(s' - s))$. The secret defined by $\mathbf{s}'$ equals s', while on the other hand $\mathbf{s}'_A = \mathbf{s}_A$. Hence, the assignment $\mathbf{b}' = \mathbf{b} + \boldsymbol{\kappa}(s' - s)$ provides a bijection between the random coins (group elements)

---

[4] Since we consider only *monotone* span programs, we omit the word "monotone".

[5] As a side remark we note that, alternatively, one could also define a span program $\mathcal{M}$ for a *pair* $(\Gamma, \mathcal{A})$ consisting of an access structure Γ and an adversary structure $\mathcal{A} \subseteq \Gamma^c$ (not necessarily $\mathcal{A} = \Gamma^c$) by requiring $\mathcal{M}$ to accept the sets in Γ and to reject the sets in $\mathcal{A}$ (while there is no condition on the sets that are neither in Γ nor in $\mathcal{A}$). This would lead, as can be easily seen from the rest of this section, to a generalized notion of secret sharing in the sense that the bound between the player sets that can reconstruct the secret (the sets in Γ) and those that have no information about it (the sets in $\mathcal{A}$) is not tight. Furthermore, it would lead to more general necessary conditions on $(\Gamma, \mathcal{A})$ for secure MPC [23, 35], in contrast to the Q^2 and Q^3 conditions considered here.

consistent with $\mathbf{s}_A$ and s and those consistent with $\mathbf{s}_A$ and s'. Therefore, $\mathbf{s}_A$ gives no Shannon information about s. This implies (perfect) privacy.

Note that since every Abelian group G is a $\mathbb{Z}$-module, an *ISP* gives rise to a *black-box* secret sharing scheme [16]. Furthermore, the above applies in particular to (the additive group of) a ring R which can be seen as a Λ-algebra.

3.3 Multiplicative Span Programs and Secure MPC

The multiplication property for a span program over a field has been introduced in [14]. It essentially requires that the product of two shared secrets can be written as a linear combination of locally computable products of shares. However, in our setting (where, given the span program, it is not clear from what ring R the secret and the shares will be sampled), we define the multiplication property as a sole property of the span program.

Let Λ be a commutative ring with 1, and let $\mathcal{M} = (\Lambda, M, \psi, \boldsymbol{\varepsilon})$ be a span program over Λ for an adversary structure $\mathcal{A}$ (i.e. $\mathcal{M}$ rejects the sets $A \in \mathcal{A}$ and accepts the sets $A \notin \mathcal{A}$).

Definition 3. *The span program $\mathcal{M}$ is called* multiplicative *if there exists a block-diagonal matrix $D \in \Lambda^{d \times d}$ such that $M^T D M = \boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T$, where block-diagonal is to be understood as follows. Let the rows and columns of D be labeled by ψ, then the non-zero entries of D are collected in blocks $D_1, \ldots, D_n$ such that for every $P_i \in \mathcal{P}$ the rows and columns in D_i are labeled by P_i.*
$\mathcal{M}$ is called strongly multiplicative *if, for every player set $A \in \mathcal{A}$, $\mathcal{M}$ restricted to the complement A^c of A is multiplicative.*

As in the case of span programs over fields (see [14]), for every adversary structure $\mathcal{A}$ there exists a (strongly) multiplicative span program $\mathcal{M}$ over Λ for $\mathcal{A}$ if and only if $\mathcal{A}$ is Q^2 (Q^3), meaning that no two (three) sets of $\mathcal{A}$ cover the whole player set $\mathcal{P}$ [27]. Furthermore, there exists an efficient procedure to transform any span program $\mathcal{M}$ over Λ for a Q^2 adversary structure $\mathcal{A}$ into a *multiplicative* span program $\mathcal{M}'$ (over Λ) for the same adversary structure $\mathcal{A}$, such that the size of $\mathcal{M}'$ is at most twice the size of $\mathcal{M}$.[6]

Similarly to the field case, the multiplication property allows to securely compute a sharing of the product of two shared secrets. Indeed, let R be a finite ring which can be seen as a Λ-algebra, and let $\mathbf{s} = M\mathbf{b}$ and $\mathbf{s}' = M\mathbf{b}'$ be sharings of two secrets $s, s' \in R$. Then, the product ss' can be written as

$$ss' = \mathbf{b}^T \boldsymbol{\varepsilon}\, \boldsymbol{\varepsilon}^T \mathbf{b}' = \mathbf{b}^T M^T D M \mathbf{b}' = (M\mathbf{b})^T D M \mathbf{b}' = \mathbf{s}^T D \mathbf{s}' = \sum_{i=1}^{n} \mathbf{s}_i^T D_i \mathbf{s}_i' \,,$$

i.e., by the special form of D, as the sum of locally computable values. Hence the multiplication protocol from [25] can be applied: To compute securely a sharing $\mathbf{s}'' = M\mathbf{b}''$ of the product ss', every player P_i shares $p_i = \mathbf{s}_i^T D_i \mathbf{s}_i'$, and then every player P_i adds up its shares of $p_1, \ldots, p_n$, resulting in P_i's share $\mathbf{s}_i''$ of ss'.

Given a multiplicative span program over Λ for a Q^2 adversary structure $\mathcal{A}$ (where the multiplication property can always be achieved according to a remark above), it follows that if R is a Λ-algebra, then any circuit over R can be computed securely with respect to a *passive* adversary that can (only) eavesdrop the players of an arbitrary set $A \in \mathcal{A}$. Namely, every player shares its private input(s) using the secret sharing scheme described in Section 3.2, and then the circuit is securely evaluated gate by gate, the addition gates non-interactively based on the homomorphic property of the secret sharing scheme, and the multiplication gates using the above mentioned multiplication protocol. Finally, the (shared) result of the computation is reconstructed. We sketch in Section 3.4 how to achieve security against an active Q^3 adversary. Note that a broadcast channel can be securely implemented using, e.g., [22]. All in all, this proves

[6] A similar result concerning the *strong* multiplication property is not known to exist, not even in the field case.

Theorem 1. *Let Λ be a commutative ring with* 1*, and let $\mathcal{M}$ be a (strongly)*[7] *multiplicative span program over Λ for a Q^3 adversary structure $\mathcal{A}$. Then there exists an $\mathcal{A}$-secure MPC protocol to evaluate any arithmetic circuit C over an arbitrary finite ring R which can be seen as a Λ-algebra.*

Concerning efficiency, the communication complexity of the MPC protocol (in terms of the number of ring elements to be communicated) is polynomial in n, in the size of $\mathcal{M}$, and in the number of multiplication gates in C.

Corollary 1. *Let $\mathcal{M}$ be a (strongly) multiplicative* ISP *for a Q^3 adversary structure $\mathcal{A}$. Then there exists an $\mathcal{A}$-secure* black-box *MPC protocol to evaluate any arithmetic circuit C over an arbitrary finite ring R.*

The black-box MPC result from Corollary 1 exploits the fact that every ring R is a $\mathbb{Z}$-algebra.[8] If, however, additional information about R is given, it might be possible to view R as an algebra over another commutative ring Λ with 1. For example, if the exponent ℓ of (the additive group of) R is given[9], then we can exploit the fact that R is an algebra over $\Lambda = \mathbb{Z}_\ell$. In many cases, this leads to smaller span programs and thus to more efficient MPC protocols than in the black-box case. For instance, if the exponent of R is a prime p then R is an algebra over the field $\mathbb{F}_p$, and we can apply standard techniques to derive span programs over $\mathbb{F}_p$ (or an extension field). If the exponent ℓ is not prime but, say, square-free, we can use Chinese Remainder Theorem to construct suitable span programs. See also Proposition 1 for the case of a threshold adversary structure.

3.4 Achieving Security Against an Active Adversary

Following the paradigm of [14], security against an *active* adversary can be achieved by means of a *linear distributed commitment* and three corresponding auxiliary protocols: a *commitment transfer protocol* (CTP), a *commitment sharing protocol* (CSP) and a *commitment multiplication protocol* (CMP). A linear distributed commitment allows a player to commit to a secret, however, in contrast to its cryptographic counterpart, a distributed commitment is perfectly hiding *and* binding. A CTP allows to transfer a commitment for a secret from one player to another, a CSP allows to share a committed secret in a verifiable way such that the players will be committed to their shares, and a CMP allows to prove that three committed secrets s, s' and s'' satisfy the relation $s'' = ss'$, if this is indeed the case. These protocols allow to modify the passively secure MPC protocol, sketched in Section 3.3, in such a way that at every stage of the MPC every player is committed to its current intermediary results. This guarantees detection of dishonest behaviour and thus security against an *active* adversary. It is straightforward to verify that the field based solutions of [14] can be extended to our more general setting of MPC over an arbitrary ring (see Appendix B). As in [14], the *perfectly* secure CMP requires a *strongly multiplicative* span program whereas an ordinary span program suffices for *unconditional* security.

3.5 Threshold Black-Box MPC

Consider a *threshold* adversary structure $\mathcal{A}_{t,n} = \{A \subseteq \mathcal{P} : |A| \leq t\}$ with $0 < t < n$.

Proposition 1. *Let Λ be a commutative ring with* 1*. Assume there exist units $\omega_1, \ldots, \omega_n \in \Lambda$ such that all pairwise differences $\omega_i - \omega_j$ $(i \neq j)$ are invertible as well. Then there exists a span program $\mathcal{M} = (\Lambda, M, \psi, \boldsymbol{\varepsilon})$ for $\mathcal{A}_{t,n}$ of size n, which is (strongly) multiplicative if and only if $t < n/2$ $(t < n/3)$: the i-th row of M is simply $(1, \omega_i, \omega_i^2, \ldots, \omega_i^t)$, labelled by P_i, and $\boldsymbol{\varepsilon} = (1, 0, \ldots, 0)^T \in \Lambda^{t+1}$.*

[7] Perfect security requires a strongly-multiplicative span program, while an (ordinary) multiplicative span program is sufficient for unconditional security (see Section 3.4).

[8] Note that the corresponding number multiplication can *efficiently* be computed using standard "double and add", requiring only black-box access to the addition in R.

[9] I.e., the smallest integer $\ell > 0$ such that $\ell \cdot a = 0$ for all $a \in R$.

The resulting secret sharing scheme (with the secret and shares sampled from a Λ-module G), formally coincides with the well known Shamir scheme [39], except that the interpolation points $\omega_1, \ldots, \omega_n$ have to be carefully chosen (from Λ). The security of this generalized Shamir scheme has been proven in [19, 18]. The proof below also includes the claim concerning the (strong) multiplication property.

Proof. Let $\mathcal{M} = (\Lambda, M, \psi, \boldsymbol{\varepsilon})$ be as suggested in Proposition 1, and let $A = \{P_{i_1}, \ldots, P_{i_{t+1}}\} \subseteq \mathcal{P}$ be a subset of cardinality $t+1$. Then, the matrix M_A is a Vandermonde matrix whose determinant is known to be $\det(M_A) = \prod_{j>k}(\omega_{i_j} - \omega_{i_k})$. Hence, by the choice of the ω_i's, $\det(M_A)$ is a unit and thus M_A is invertible. This implies that $\boldsymbol{\varepsilon} \in \mathrm{im}(M_A^T)$, as required.
Let now $A = \{P_{i_1}, \ldots, P_{i_t}\} \subseteq \mathcal{P}$ be a subset of cardinality t, and let $N_A \in \Lambda^{t\times t}$ denote the matrix consisting of the 2nd up to the last column of M_A. It is not hard to see that $\det(N_A) = \omega_{i_1} \cdots \omega_{i_t} \prod_{j>k}(\omega_{i_j} - \omega_{i_k})$. This implies, again by the choice of the ω_i's, that the first column of M_A is in the image of N_A. This, however, is equivalent to the existence of a vector $\boldsymbol{\kappa}$ with $M_A\boldsymbol{\kappa} = \mathbf{0}$ and $\kappa_1 = 1$, which had to be shown.

Concerning the multiplication property, if $D \in \Lambda^{n\times n}$ is a diagonal matrix with diagonal $(d_1, \ldots, d_n)$ then $M^T DM$ is of the form

$$M^T DM = \sum_i d_i \begin{pmatrix} 1 & \omega_i & \cdots & \omega_i^t \\ \omega_i & \omega_i^2 & \cdots & \omega_i^{t+1} \\ \vdots & \vdots & & \vdots \\ \omega_i^t & \omega_i^{t+1} & \cdots & \omega_i^{2t} \end{pmatrix}$$

By the above, we know that if $2t < n$ then there exist $d_1, \ldots, d_n$ such that $\sum_i d_i(1, \omega_i, \ldots, \omega^{2t}) = (1, 0, \ldots, 0)$, and hence, choosing the diagonal of D this way, it holds that $M^T DM = \boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T$. The strong multiplication property can be shown similarly in case $3t < n$. $\square$

To achieve black-box MPC over an arbitrary finite ring R, it suffices, by Corollary 1, to have a (strongly multiplicative) ISP for the considered adversary structure $\mathcal{A}_{t,n}$. Unfortunately, the ring $\Lambda = \mathbb{Z}$ does *not* fulfill the assumption of Proposition 1 (except for $n = 1$), and hence Proposition 1 does not provide the desired ISP. However, by Lemma 1 below, it is in fact sufficient to provide a span program over an *extension ring* Λ of $\mathbb{Z}$, as it guarantees that any such span program can be "projected" to an ISP.[10] The remaining gap is then closed in Lemma 2 by exhibiting an extension ring Λ of $\mathbb{Z}$ that satisfies the assumption of Proposition 1.

Lemma 1. *Let $f(X) \in \mathbb{Z}[X]$ be a monic, irreducible polynomial of non-zero degree m, and let Λ be the extension ring $\Lambda = \mathbb{Z}[X]/(f(X))$ of $\mathbb{Z}$. Then, any span program $\mathcal{M}$ over Λ can be (efficiently) transformed into an* integer *span program $\bar{\mathcal{M}}$ for the same adversary structure such that $\mathrm{size}(\bar{\mathcal{M}}) = m \cdot \mathrm{size}(\mathcal{M})$. Furthermore, if $\mathcal{M}$ is (strongly) multiplicative then this also holds for $\bar{\mathcal{M}}$.*

The first part of this lemma appeared in [16]. A full proof of Lemma 1, also covering the multiplication property, is given in Appendix A. For a proof of Lemma 2 below we refer to [19].

Lemma 2. *Consider the polynomials $\omega_i(X) = 1 + X + \cdots + X^{i-1} \in \mathbb{Z}[X]$ for $i = 1, \ldots, n$. Then $\omega_1(X), \ldots, \omega_n(X)$ and all pairwise differences $\omega_i(X) - \omega_j(X)$ $(i \neq j)$ are invertible modulo the cyclotomic polynomial $\Phi_q(X) = 1 + X + \cdots + X^{q-1} \in \mathbb{Z}[X]$, where q is a prime greater than n.*

[10] Alternatively, one could also "lift" R to an extension ring $S \supseteq R$ which can be seen as an algebra over $\Lambda \supseteq \mathbb{Z}$, and then do the MPC over S, using some mechanism that ensures that the inputs come from the smaller ring R. This approach, which has also been used in [19] in the context of secret sharing over arbitrary Abelian groups, would lead to a somewhat more efficient implementation of the MPC protocols; however, we feel that our approach serves better conceptual simplicity.

Hence, if $t < n/3$ then by Proposition 1 there exists a strongly-multiplicative span program $\mathcal{M}$ for the threshold adversary structure $\mathcal{A}_{t,n}$ over the extension ring $\Lambda = \mathbb{Z}[X]/(\Phi_q(X))$ where $q > n$. The size of $\mathcal{M}$ is n, and q can be chosen linear in n by Bertrand's Postulate. Together with Lemma 1, this implies a strongly-multiplicative ISP of size $O(n^2)$, and hence Corollary 1 yields

Corollary 2. *For $t < n/3$, there exists an $\mathcal{A}_{t,n}$-secure black-box MPC protocol to evaluate any arithmetic circuit C over an arbitrary finite ring R.*

A span program for $\mathcal{A}_{t,n}$ of size $2n$ over an extension ring Λ of $\mathbb{Z}$ of degree *logarithmic* in n was presented in [16], leading to an ISP of size $O(n \log n)$. As this construction too is related to Shamir's scheme, it is not hard to see that also this ISP is (strongly) multiplicative if and only if $t < n/2$ ($t < n/3$). Hence, it gives rise to another instantiation of the MPC protocol claimed in Corollary 2. It turns out that using a broadcast protocol with optimal message complexity $O(n^2)$ (but $O(n)$ rounds), e.g. [6], this MPC protocol requires $O(|C|n^6 \log n)$ elements of R to be communicated, where $|C|$ denotes the number of multiplication gates in C. Hence, the communication complexity coincides asymptotically with the classical protocols of [7, 2, 24], up to a possible loss of a factor $\log n$, which is due to the fact that over *large* fields there exist threshold span programs of size n. Furthermore, our protocol is compatible with improvements to the communication complexity of non-black-box MPC over fields [29, 28].

4 Constant-Round Protocols

In this section we present constant-round MPC protocols over arbitrary rings. Our motivation is twofold. First, we complement the results of the previous section by showing that they carry over in their full generality to the constant-round setting. This does not immediately follow from previous work in the area. Second, we point out some improvements and simplifications to previous constant-round techniques, which also have relevance to the special case of fields. In particular, we obtain constant-round protocols for small-width branching programs and balanced formulas in which the communication complexity is nearly linear in their size.

4.1 Randomizing Polynomials over Rings

The results of the previous section may be viewed as providing a general "compiler", taking a description of an arithmetic circuit C over some ring R and producing a description of an MPC protocol for the functionality prescribed by C. While the communication complexity of the resultant protocol is proportional to the size of C, its round complexity is proportional to its *multiplicative depth.*[11] In particular, constant-degree polynomials over R can be securely evaluated in a constant number of rounds using black-box access to R. The notion of *randomizing polynomials*, introduced in [30], provides a convenient framework for representing complex functions as low-degree polynomials, thereby allowing their round-efficient secure computation. In the following we generalize this notion to apply to any function $f : R^n \to D$, where R is an arbitrary ring and D is an arbitrary set.[12]

A *randomizing polynomials vector* over the ring R is a vector $p = (p_1, \ldots, p_s)$ of s multivariate polynomials over R, each acting on the same $n + m$ variables $x = (x_1, \ldots, x_n)$ and $r = (r_1, \ldots, r_m)$. The variables x are called *inputs* and r are called *random inputs.* The *complexity* of p is the total number of inputs and outputs (i.e., $s + n + m$). Its *degree* is defined as the maximal degree of its

[11] Multiplicative depth is defined similarly to ordinary circuit depth, except that addition gates and multiplications by constant do not count.

[12] In this section the parameter n is used to denote an input length parameter rather than the number of players. The input, taken from R^n, may be arbitrarily partitioned among any number of players.

s entries, where both ordinary inputs and random inputs (but not constants) count towards the degree.[13]

Representation of a function f by p is defined as follows. For any $x \in R^n$, let $P(x)$ denote the *output distribution* of $p(x,r)$, induced by a uniform choice of $r \in R^m$. Note that for any input x, $P(x)$ is a distribution over s-tuples of ring elements. We say that p *represents* a function f if the output distribution $P(x)$ is "equivalent" to the function value $f(x)$. This condition is broken into two requirements, *correctness* and *privacy*, as formalized below.

Definition 4. *A randomizing polynomials vector $p(x,r)$ is a said to* represent *the function $f : R^n \to D$ if the following requirements hold:*

- Correctness. *There exists an efficient*[14] *reconstruction algorithm which, given only a sample from $P(x)$, can correctly compute the output value $f(x)$.*
- Privacy. *There exists an efficient* simulator *which, given the output value $f(x)$, can emulate the output distribution $P(x)$.*

*We will also consider the relaxed notion of δ-*correct *randomizing polynomials, where the reconstruction algorithm is allowed to output "don't know" with probability δ (but otherwise must be correct).*

The application of randomizing polynomials to secure computation, discussed in [30], is quite straightforward. Given a representation of $f(x)$ by $p(x,r)$, the secure computation of f can be reduced to the secure computation of the randomized function $P(x)$. The latter, in turn, reduces to the secure computation of the *deterministic* function $p'(x, r^1, \ldots, r^a) \stackrel{\text{def}}{=} p(x, r^1 + \ldots + r^a)$, where a is the size of some set $A \notin \mathcal{A}$, by assigning each input vector r^j to a distinct player in A and instructing it to pick r^j at random. Note that the degree of p' is the same as that of p. Moreover, if the reconstruction procedure associated with p requires only black-box access to R, then this property is maintained by the reduction. Hence, using the results of the previous section, the problem of getting round-efficient MPC over rings reduces to that of obtaining low-degree representations for the functions of interest.

In the following we describe two constructions of degree-3 randomizing polynomials over rings, drawing on techniques from [31, 3].

4.2 Branching Programs over Rings

Branching programs are a useful and well-studied computational model. In particular, they are stronger than the formula model (see Section 4.3). We start by defining a general notion of branching programs over an arbitrary ring R.

Definition 5. *A branching program BP on inputs $x = (x_1, \ldots, x_n)$ over R is defined by: (1) a DAG (directed acyclic graph) $G = (V, E)$; (2) a weight function w, assigning to each edge a degree-1 polynomial over R in the input variables. It is convenient to assume that $V = \{0, 1, \ldots, \ell\}$, where ℓ is referred to as the* size *of BP, and that for each edge $(i,j) \in E$ it holds that $i < j$. The function computed by BP is defined as follows. For each directed path $\phi = (i_1, i_2, \ldots, i_k)$ in G, the weight of ϕ is defined to be the product $w(i_1, i_2) \cdot w(i_2, i_3) \cdot \ldots \cdot w(i_{k-1}, i_k)$ (in the prescribed order). For $i < j$, we denote by $W(i,j)$ the total weight of all directed paths from i to j (viewed as a function of x). Finally, the function $f : R^n \to R$ computed by BP is defined by $f(x) = W(0, \ell)(x)$. We refer to $W(0, \ell)$ as the* output *of BP.*

Note that, using a simple dynamic programming algorithm, the output of BP can be evaluated from its edge weights using $O(|E|)$ black-box ring operations.

[13] It is crucial for the MPC application that random inputs count towards the degree.

[14] The efficiency requirement can only be meaningfully applied to a *family* of randomizing polynomials, parameterized by the input size n and the ring R.

To represent a branching program by randomizing polynomials, we rely on a recent construction from [31].[15] However, applying this construction to general rings requires a different analysis. In particular, the original analysis relies on properties of the determinant which do not hold in general over non-commutative rings. Below we provide a more combinatorial interpretation of this construction, which may be of independent interest.

How to garble a branching program. Given a branching program $BP = (G, w)$ of size ℓ, we define a randomized procedure producing a "garbled" branching program $\tilde{BP} = (\tilde{G}, \tilde{w})$ of the same size ℓ. The graph $\tilde{G}$ will always be the complete DAG, i.e., each (i, j) where $0 \le i < j \le \ell$ is an edge in $\tilde{G}$. We will sometimes also view G as a complete graph, where $w(i, j) = 0$ if (i, j) is not originally an edge. The randomization of BP proceeds in two phases.

Main phase. Let r_{ij}, $0 \le i < j < \ell$, be $\binom{\ell}{2}$ random and independent ring elements. Each r_{ij} naturally corresponds to an edge. The main randomization phase modifies each original weight $w(i, j)$ as follows: first, if $j < \ell$, it increases it by r_{ij}. Then, regardless of whether $j = \ell$, it decreases it by $r_{ih} \cdot w(h, j)$ for each h lying strictly between i and j. That is, the updated weights $w'(i, j)$ obtained at the end of this phase are defined by

$$w'(i,j) = \begin{cases} w(i,j) + r_{ij} - \sum_{h=i+1}^{j-1} r_{ih} \cdot w(h,j), & j < \ell \\ w(i,j) - \sum_{h=i+1}^{j-1} r_{ih} \cdot w(h,j), & j = \ell \end{cases}$$

Note that each $w'(i, j)$ is a degree-2 polynomial in the inputs x and the random inputs r_{ij}.

Cleanup phase. In the main phase the weights of the edges entering ℓ were not fully randomized. In some sense, these edges served as "garbage collectors" for the randomization of the remaining edges. To eliminate the unwanted residual information about x, the following operation is performed. Let $r'_1, \ldots, r'_{\ell-1}$ be independent random ring elements. The new weights $\tilde{w}$ are the same as w' for (i, j) such that $j < \ell$, and else are defined by:

$$\tilde{w}(i,\ell) = \begin{cases} w'(i,\ell) - \sum_{j=i+1}^{\ell-1} w'(i,j) \cdot r'_j, & i = 0 \\ w'(i,\ell) + r'_i - \sum_{j=i+1}^{\ell-1} w'(i,j) \cdot r'_j, & i > 0 \end{cases}$$

Note that the weights $\tilde{w}(i, \ell)$ are degree-3 polynomials in x, r, r' and the remaining weights are all of degree 2. Still, each weight $\tilde{w}$ is of degree 1 in x, and hence any fixed choice of r, r' indeed makes $\tilde{BP}$ a branching program according to our definition.

We define a randomizing polynomials vector $p(x, r, r')$ representing BP by the concatenation of all $\binom{\ell+1}{2}$ weights $\tilde{w}$. It has degree 3 and complexity $O(\ell^2)$. It can be evaluated using $O(|E|\ell)$ ring operations assuming that each original weight w depends on a single input variable. We now prove its correctness and privacy.

Correctness. It suffices to show that on any input x, the value of $\tilde{BP}$ equals that of BP, for any choice of r, r'. For this, it suffices to show that the positive and negative contributions of each random input cancel each other. Consider the effect of a specific random input r_{ij} in the main phase. It is involved in two types of operations: (1) it is added to $w(i, j)$; and (2) $r_{ij} \cdot w(j, k)$ is subtracted from each weight $w(i, k)$ such that $k > j$. We now compare the contribution of (1) and (2) to the output $W(0, \ell)$. Since (1) affects exactly those paths that traverse the edge (i, j), the positive contribution of (1) is

$$W(0, i) \cdot r_{ij} \cdot W(j, \ell).$$

[15] The goal of [31] is to obtain perfectly-secure MPC protocols with a constant number of rounds in the *worst case*. The fact that the same construction is useful in the current context as well may be viewed as a fortunate coincidence.

(Note that, by the distributive law, the above expression covers exactly all directed paths from 0 to ℓ passing through (i,j).) Similarly, the negative contribution of (2) is:

$$\begin{aligned}\sum_{k>j} W(0,i)\cdot(r_{ij}\cdot w(j,k))\cdot W(k,\ell) &= W(0,i)\cdot r_{ij}\sum_{k>j} w(j,k)\cdot W(k,\ell)\\ &= W(0,i)\cdot r_{ij}\cdot W(j,\ell)\end{aligned}$$

Hence, the positive and negative contributions of each r_{ij} exactly cancel each other, as required. A similar argument applies to the cleanup phase operations involving r'_i (details omitted). To conclude, it suffices for the reconstruction procedure to evaluate the garbled branching program $\tilde{BP}(x,r,r')$, which requires $O(\ell^2)$ ring operations.

Privacy. We argue that, for any fixed x, the distribution of $\tilde{w}$ induced by the random choice of r,r' is uniform among all weight assignments having the same output value as BP on x. First, note that the number of possible choices of r,r' is $|R|^{\binom{\ell}{2}+(\ell-1)}$, which is exactly equal to $|R|^{\binom{\ell+1}{2}-1}$, the number of possible weight assignments in each output class.[16] It thus suffices to prove that, for any fixed weight assignment w, the effect of r,r' on w (as a function from $R^{\binom{\ell+1}{2}-1}$ to $R^{\binom{\ell+1}{2}}$) is one-to-one. Consider two distinct vectors of random inputs, (r,r') and $(\hat{r},\hat{r}')$. Order each of them by first placing the r_{ij} entries in increasing lexicographic order and then the r'_i entries in decreasing order. Consider the first position where the two ordered lists differ. It is not hard to verify that if the first difference is $r_{ij}\neq\hat{r}_{ij}$, where $j<\ell$, then the weight of (i,j) will differ after the main phase. (Note that since $j<\ell$, this weight is untouched in the cleanup phase.) The second case, where the first difference is $r'_i\neq\hat{r}'_i$, is similar. In this case the two random inputs will induce the same change to the weight of (i,ℓ) in the main phase, and a different change in the cleanup phase. Thus, the garbled weight function is indeed uniformly random over its output class. Given the above, a simulator may proceed as follows. On output value $d\in R$, the simulator constructs a branching program BP with $w(0,\ell)=d$ and $w(i,j)=0$ elsewhere, and outputs a garbled version $\tilde{BP}$ of BP.

Combining the above with the results of the previous section, we have:

Theorem 2. *Let BP be a branching program over a black-box ring R, where BP has size ℓ and m edges. Then BP admits a perfectly secure MPC protocol, communicating $O(\ell^2)$ ring elements and performing $O(m\ell)$ ring operations (ignoring polynomial dependence on the number of players). The protocol may achieve an optimal security threshold, and its exact number of rounds corresponds to that of degree-3 polynomials.*

TRADING COMMUNICATION FOR ROUNDS. For large branching programs, the quadratic complexity overhead of the previous construction may be too costly. While this overhead somehow seems justified in the general case, where the description size of BP may also be quadratic in its size ℓ, one can certainly expect improvement in the typical case where BP has a sparse graph. A useful class of such branching programs are those that have a small *width*. BP is said to have length a and width b if the vertices of its graph G can be divided into a levels of size $\leq b$ each, such that each edge connects two consecutive levels. For instance, for any binary regular language, the words of length n can be recognized by a constant-width length-n branching program over $\mathbb{Z}_2$ (specifically, the width is equal to the number of states in the corresponding automaton).

For the case of small-width branching programs, we can almost eliminate the quadratic overhead at the expense of a moderate increase in the round complexity. We use the following recursive decomposition approach. Suppose that the length of BP is broken into s segments of length a/s each. Moreover, suppose that in each segment all b^2 values $W(i,j)$ such that i is in the first level

[16] Indeed, among the $|R|^{\binom{\ell+1}{2}}$ possible ways of fixing the weights, there is an equal representation for each output. A bijection between the weight functions of two output values d_1,d_2 can be obtained by adding d_1-d_2 to the weight of $(0,\ell)$.

of that segment and j is in the last level are evaluated. Then, it is not hard to see that the output of BP can be computed by a branching program BP' of length s and width b, such that the edge weights of BP' are the $b^2 s$ weights $W(i,j)$ as above. Thus, the secure computation of BP can be broken into two stages: first evaluate in parallel $b^2 s$ branching programs[17] of length a/s and width b each, producing the weights $W(i,j)$, and then use these weights as inputs to the length-s, width-b branching program BP', producing the final output. This process requires to hide the intermediate results produced by the first stage, which can be done with a very small additional overhead. In fact, a careful implementation (following [3]) allows the second stage to be carried out using only a single additional round of broadcast.

If the width b of BP is constant, an optimal choice for s is $s = O(a^{2/3})$, in which case the communication complexity of each of the two stages becomes $O(a^{4/3})$. This is already a significant improvement over the $O(a^2)$ complexity given by Theorem 2. Moreover, by recursively repeating this decomposition and tuning the parameters, the complexity can be made arbitrarily close to linear while maintaining a (larger) constant number of rounds. In particular, this technique can be used to obtain nearly-linear perfect constant-round protocols for iterated ring multiplication or for Yao's millionaires' problem [40], both of which admit constant-width linear-length branching programs.

4.3 Arithmetic Formulas

An *arithmetic formula* over a ring R is defined by a rooted binary tree, whose leaves are labeled by input variables and constants (more generally, by degree-1 polynomials), and whose internal nodes, called *gates*, are labeled by either '+' (addition) or '×' (multiplication). If R is non-commutative, the children of each multiplication node must be ordered. A formula is evaluated in a gate-by-gate fashion, from the leaves to the root. Its *size* is defined as the number of leaves and its *depth* as the length of the longest path from the root to a leave. A formula is *balanced* if it forms a complete binary tree.

We note that the branching program model is strictly stronger than the formula model. In particular, any formula (even with gates of unbounded fan-in) can be simulated by a branching program of the same size. Thus, the results from Section 4.2 apply to formulas as well.

We combine a complexity result due to Cleve [13] with a variant of a randomization technique due to Beaver [3] (following Kilian [33] and Feige et al. [21]) to obtain an efficient representation of formulas by degree-3 randomizing polynomials. If the formula is balanced, the complexity of this representation can be made nearly linear in the formula size. However, in contrast to the previous construction, the current one will not apply to a *black-box* ring R and will not offer perfect correctness. Still, for the case of balanced arithmetic formulas, it can provide better efficiency.

From formulas to iterated matrix product. In [13], it is shown that an arithmetic formula of depth d over an arbitrary ring R with 1 can be reduced to the *iterated product* of $O((2^d)^{1+\frac{2}{b}})$ matrices of size $c = O(2^b)$ over R, for any constant b. Each of these matrices is invertible, and its entries contain only variables or constants.[18] The output of the formula is equal to the top-right entry of the matrix product. Note that if the formula is balanced, then the total size of all matrices can be made nearly linear in the formula size.

Next, we consider the problem of randomizing an entry of an iterated matrix product as above. Having already established the possibility of constant-round MPC over black-box rings, we focus on efficiency issues and restrict our attention to the special case of fields. Indeed, the following construction does not apply to black-box rings (though may still apply with varied efficiency to non-field rings). In what follows we let K denote a finite field, $K^{c\times c}$ the set of $c \times c$ matrices over K, and $\mathrm{GL}_c(K)$ the group of *invertible* $c \times c$ matrices over K.

[17] It is possible to avoid the b^2 overhead by modifying the garbled branching program construction so that all weights $W(i,j)$ in each segment are evaluated at once.

[18] The requirement that R has 1 can be dispensed with by using an appropriate extension of R.

RANDOMIZING AN ITERATED PRODUCT OF INVERTIBLE MATRICES. Kilian [33], in the context of secure *two-party* computation, suggested the following approach for randomizing an iterated group product (formulated here for $\mathrm{GL}_c(K)$). Given k matrices $M_1, M_2, \ldots, M_k \in \mathrm{GL}_c(K)$, learning their iterated product $M_1 M_2 \cdots M_k$ is equivalent to learning the k-tuple

$$(M_1 S_1\,,\ S_1^{-1} M_2 S_2\,,\ S_2^{-1} M_2 S_3, \ldots, S_{k-2}^{-1} M_{k-1} S_{k-1}\,,\ S_{k-1}^{-1} M_k), \tag{1}$$

where $S_1, \ldots, S_{k-1}$ are independent, random matrices from $\mathrm{GL}_c(K)$. The k-tuple of randomized matrices produced by (1) are uniformly random subject to the requirements that they are all invertible and their product is equal to the original product. This approach was adapted to the multi-party setting in [1], where special-purpose MPC protocols for generating secret invertible matrices and for inverting matrices were devised. To eliminate the interaction required for matrix inversion, Beaver [3] suggested to modify (1) by randomizing $M_1 \cdots M_k$ as:

$$(M_1 S_1\,,\ S_2 S_1\,,\ S_2 M_2 S_3\,,\ S_4 S_3\,,\ S_4 M_3 S_5, \ldots, S_{2k-2} S_{2k-3}\,,\ S_{2k-2} M_k), \tag{2}$$

where $S_1, \ldots, S_{2k-2}$ are again random invertible matrices. It is easy to verify that the original product can be recovered from (2) by first inverting every other matrix and then multiplying the $2k-1$ matrices (in fact, (2) may be obtained by first applying (1) to the product $M_1 I M_2 I \cdots M_{k-1} I M_k$ and then inverting every other matrix). Note that both (1) and (2) reveal all entries of the product matrix, whereas we would like to reveal only the top-right entry and eliminate all additional information. To get around a similar problem, Feige et al. [21] construct two carefully chosen distributions on $c \times c$ matrices, D_L, D_R, such that $D_L M D_R$ reveals only the top-right entry of M. This procedure was used in [21, 3] to randomize nondeterministic branching programs, and applies in our context as well. Thus, by applying (1) or (2) to the product $D_L M_1 \cdots M_k D_R$, only the top-right entry of $M_1 \cdots M_k$ is revealed.

While (2) eliminates the need for secure matrix inversion, it still requires matrix inversion for reconstructing the original product. This operation cannot be applied for general matrices over a black-box ring. Moreover, generating the invertible matrices S_i still requires a separate subprotocol which incurs additional interaction.

We suggest two modifications to the above randomization technique. First, we convert it to the randomizing polynomials framework, by using totally random matrices S_i instead of invertible ones. This allows to eliminate the additional interaction required for generating secret invertible matrices. It is clear that when the fraction of $c \times c$ matrices over R that are singular is small, this modification will have little effect on the privacy or the correctness. We show that, in fact, *perfect* privacy is always maintained. However, for the error probability δ to be small, it is still required that the fraction of invertible matrices be large, and in particular R must be large. This is another "non-black-box" aspect of the current approach. (A similar problem arises if an interactive protocol for generating invertible matrices is used, as in [1, 3].)

A second modification is the use of a simpler and more efficient alternative to the technique from [21] for eliminating all but one entry in the matrix product. Specifically, instead of appending the (nontrivial) matrix distributions D_L and D_R to (1) or (2), it suffices to *remove* all rows except the first in M_1, and all columns except the last in M_k. (Note that it is not clear a-priori that this modification indeed eliminates all unwanted information.) While the efficiency advantage gained by this simplification is quite minor in the current context, it may be more significant in others. For instance, it allows to securely compute the inner-product of two n-vectors in the non-interactive model of [21] with $O(n)$ communication complexity (rather than $O(n^2)$).[19]

The above discussion is captured by the following proposition.

[19] This is achieved by reducing the inner product to the product of two matrices, and applying the optimized version of (1).

Proposition 2. *Let $M_2, \ldots, M_{k-1} \in \mathrm{GL}_c(K)$, let $\hat{M}_1$ be a nonzero row vector and $\hat{M}_k$ a nonzero column vector. Suppose that at most a $(\delta/2k)$-fraction of the $c \times c$ matrices over K are singular. Then,*

$$(\hat{M}_1 S_1 \ , \ S_2 S_1 \ , \ S_2 M_2 S_3 \ , \ S_4 S_3 \ , \ S_4 M_3 S_5, \ldots, S_{2k-2} S_{2k-3} \ , \ S_{2k-2} \hat{M}_k), \tag{3}$$

where $S_1, \ldots, S_{2k-2}$ are uniformly random matrices, is a δ-correct degree-3 representation for the iterated product $\hat{M}_1 M_2 \cdots M_{k-1} \hat{M}_k$.

To prove Proposition 2 we rely on the following two lemmas.

Lemma 3. *Let K be a field. For any $x, y \in K^c \setminus \{0\}$ and $M \in \mathrm{GL}_c(K)$, there exist $X, Y \in \mathrm{GL}_c(K)$ such that the first row of X is x^T, the first column of Y is y, and XMY is a canonical matrix depending only on $x^T y$.*

Proof. Consider first the case where $x^T y = c \neq 0$. Pick the remaining $c - 1$ rows of X so that they form a basis for the space $(My)^{\perp}$. Since x is not in this space, X is invertible as required. Now, fix each jth column of Y, $2 \leq j \leq c$, as the unique solution z to the equation $XMz = ce_j$, where e_j is the jth unit vector. It follows that $XMY = cI$, where cI depends only on c as required, and that Y is invertible.

The case $x^T y = 0$ is handled similarly. In this case $x \in (My)^{\perp}$. We pick the next $c-2$ rows of X so that together with x they form a basis for $(My)^{\perp}$. The last row of X is picked so that its inner product with My is 1. This guarantees that X is invertible. Now, each jth column of Y, $2 \leq j \leq c$, is fixed as the unique solution z to the equation $XMz = e_{c-j+1}$. It follows that in this case XMY is always the "mirror image" of the unit matrix. $\square$

Lemma 4. *Let $M_1, \ldots, M_k \in \mathrm{GL}_c(K)$, and let $S_1, \ldots, S_{2k-1}$ be random and independent matrices in $K^{c \times c}$. Then, the distribution of*

$$(M_1 S_1 \ , \ S_2 S_1 \ , \ S_2 M_2 S_3 \ , \ S_4 S_3 \ , \ S_4 M_3 S_5, \ldots, S_{2k-2} S_{2k-3} \ , \ S_{2k-2} M_k) \tag{4}$$

can be efficiently simulated given $M_1 M_2 \cdots M_k$.

Proof. We first describe the simulator and then prove its correctness. The simulator lets $M = M_1 M_2 \cdots M_k$, picks random and independent matrices $S'_1, \ldots, S'_{2k-2} \in K^{c \times c}$, and outputs

$$(S'_1 \ , \ S'_2 S'_1 \ , \ S'_2 S'_3 \ , \ S'_4 S'_3 \ , \ S'_4 S'_5, \ldots, S'_{2k-2} S'_{2k-3} \ , \ S'_{2k-2} M). \tag{5}$$

To see that (4) and (5) are equally distributed, let $P_i = \prod_{j=1}^{i} M_i$, and let $\sigma : \left(K^{c \times c}\right)^{2k-2} \to \left(K^{c \times c}\right)^{2k-2}$ be the bijection defined by

$$\sigma(S_1, \ldots, S_{2k-2}) = (P_1 S_1 \ , \ S_2 P_1^{-1} \ , \ P_2 S_3 \ , \ S_4 P_2^{-1} \ , \ P_3 S_5 \ , \ \ldots, \ S_{2k-2} P_{k-1}^{-1}).$$

It is easy to verify that (4) and (5) take identical values when $(S'_1, \ldots, S'_{2k-2}) = \sigma(S_1, \ldots, S_{2k-2})$. $\square$

Proof of Proposition 2: It follows from Lemma 3 that $\hat{M}_1, \hat{M}_k$ can be completed to $M_1, M_k \in \mathrm{GL}_c(K)$ such that the product $M_1 \cdots M_k$ contains no more information than (and can be simulated from) $\hat{M}_1 M_2 \cdots M_{k-1} \hat{M}_k$. The privacy part of Proposition 2 follows from Lemma 4 and from the fact that (3) is a restriction of (4). The correctness follows by noting that if all random matrices S_i are invertible, then the desired product can be reconstructed from (3) by inverting every second matrix and multiplying the matrices together. Moreover, the probability of the bad event in which some matrix S_i is singular is bounded by $(\delta/2k)(2k-2) < \delta$, and the reconstruction algorithm can recognize this event by testing whether all $k-2$ middle matrices are invertible. $\square$

Note that if the field K is small, the correctness probability can be boosted by working over an extension field (thereby increasing the probability of picking invertible matrices).

Combining Proposition 2 with Cleve's reduction, we have:

Theorem 3. *Let F be an arithmetic formula of depth d over a finite field K. Then, F admits a constant-round MPC protocol communicating $2^{d+O(\sqrt{d})}$ field elements (i.e., $s \cdot 2^{O(\sqrt{\log s})}$ elements if F is a balanced formula of size s). The protocol can either have a minimal round complexity (corresponding to degree-3 polynomials) with $O(|K|^{-1})$ failure probability, or alternatively achieve perfect correctness and privacy in an* expected *constant number of rounds (where the expected overhead to the number of rounds can be made arbitrarily small).*

5 Application: Securely Computing the Maximum Function

Aside from its theoretical value, the study of MPC over non-field rings is motivated by the possibility of embedding useful computation tasks into their richer structure. In this section we demonstrate the potential usefulness of this approach by describing an application to the round-efficient secure computation of the maximum function.

Suppose there are n players, where each player P_i holds an integer y_i from the set $\{0, 1, \dots, M\}$. (We consider M to be a feasible quantity.) Our goal is to design a protocol for securely evaluating $\max(y_1, \dots, y_n)$ with the following optimization criteria in mind. First, we would like the round complexity to be as small as possible. Second, we want to minimize the communication complexity subject to the latter requirement.

Our solution proceeds as follows. Let k be a (statistical) security parameter, and fix a ring $R = \mathbb{Z}_{Q^M}$ where Q is a k-bit prime. We denote the elements of R by $1, 2, \dots, Q^M = 0$. Consider the degree-2 randomizing polynomial

$$p(x_1, \dots, x_n, r_1, \dots, r_n) = \sum_{i=1}^{n} r_i x_i$$

over R. It is not hard to verify that: (1) the additive group of R has exactly $M + 1$ subgroups, and these subgroups are *totally* ordered with respect to containment;[20] and (2) the output distribution $P(x_1, \dots, x_n)$ is uniform over the maximal (i.e., largest) subgroup generated by an input x_i. Specifically, $P(x_1, \dots, x_n)$ is uniform over the subgroup generated by Q^j, where j is the maximal integer from $\{0, 1, \dots, M\}$ such that Q^j divides all x_i.

We are now ready to describe the protocol. First, each player i maps its input y_i to the ring element $x_i = Q^{M-y_i}$. Next, the players securely sample an element z from the output distribution $P(x_1, \dots, x_n)$. This task can be reduced to the secure evaluation of a *deterministic* degree-2 polynomial over R (see Section 4.1). Finally, the output of the computation is taken to be the index of the minimal subgroup of R containing z; i.e., the output is 0 if $z = 0$ and otherwise it is $M - \max\{\, j : Q^j \text{ divides } z\}$. Note that the value of z reveals no information about the inputs y_i except what follows from their maximum, and the protocol produces the correct output except with probability $1/Q \leq 2^{-(k-1)}$. We stress that an active adversary (or malicious players) cannot gain any advantage by picking "invalid" inputs x_i^* to the evaluation of P. Indeed, any choice of x_i^* is equivalent to a valid choice of x_i generating the same subgroup.

We turn to analyze the protocol's efficiency. Recall that our main optimization criterion was the round complexity. The protocol requires the secure evaluation of a single *degree-2* polynomial over R. Using off-the-shelf MPC protocols (adapted to the ring R as in Section 3), this requires fewer rounds than evaluating degree-3 polynomials or more complex functions.[21] The communication complexity of the protocol is linear in M and polynomial in the number of players. In Appendix C we discuss several alternative approaches for securely evaluating the maximum function. All of these alternatives

[20] This should be contrasted with the field of the same cardinality, which has 2^M *partially* ordered additive subgroups.

[21] The exact number of rounds being saved depends on the specific setting, e.g. on whether a broadcast channel is available.

either require more rounds, require a higher communication complexity (quadratic in M), or fail to remain secure against an active adversary.

Acknowledgements

We would like to thank Ivan Damgård and Tal Rabin for helpful discussions.

References

1. J. Bar-Ilan and D. Beaver. Non-cryptographic fault-tolerant computing in a constant number of rounds of interaction. In *Proc. of 8th PODC*, pp. 201-209, 1989.
2. D. Beaver. Efficient multiparty protocols using circuit randomization. In *Proc. of CRYPTO '91*, LNCS 576, pp. 420-432, 1991.
3. D. Beaver. Minimal-latency secure function evaluation. In *Proc. of EUROCRYPT '00*, LNCS 1807, pp. 335-350, 2000.
4. D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Security with low communication overhead (extended abstract). In *Proc. of CRYPTO '90*, LNCS 537, pp. 62-76, 1990.
5. D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols (extended abstract). In *Proc. of 22nd STOC*, pp. 503-513, 1990.
6. P. Berman, J. A. Garay, and K. J. Perry. Towards optimal distributed consensus (extended abstract). In *Proc. of 30th FOCS*, pp. 410-415, 1989.
7. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. of 20th STOC*, pp. 1-10, 1988.
8. R. Canetti. Security and composition of multiparty cryptographic protocols. In *J. of Cryptology*, 13(1):143-202, 2000.
9. R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *Proc. of 42nd FOCS*, pp. 136-145, 2001.
10. R. Canetti, U. Feige, O. Goldreich and M. Naor. Adaptively secure computation. In *Proc. of 28th STOC*, pp. 639-648, 1996.
11. D. Chaum, C. Crepeau, and I. Damgård. Multiparty unconditional secure protocols. In *Proc. of 20th STOC*, pp. 11-19, 1988.
12. C. Cachin, J. Camenisch, J. Kilian, and J. Muller. One-round secure computation and secure autonomous mobile agents. In *Proc. of 27th ICALP*, pp. 512-523, 2000.
13. R. Cleve. Towards Optimal Simulations of Formulas by Bounded-Width Programs. In *Computational Complexity* 1: 91-105, 1991.
14. R. Cramer, I. Damgård, and U. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *Proc. of EUROCRYPT '00*, LNCS 1807, pp. 316-334, 2000.
15. R. Cramer, I. Damgård, and J. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Proc. of EUROCRYPT '01*, LNCS 2045, pp. 280-299, 2001.
16. R. Cramer and S. Fehr. Optimal black-box secret sharing over arbitrary Abelian groups. In *Proc. of CRYPTO '02*, LNCS 2442, 272-287, 2002.
17. R. Cramer, S. Fehr, Y. Ishai, and E. Kushilevitz. Efficient multi-party computation over rings. In *Proc. of EUROCRYPT '03*, LNCS, 2003.
18. A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely. In *Proc. of 26th STOC*, pp. 522-533, 1994.
19. Y. G. Desmedt and Y. Frankel. Homomorphic zero-knowledge threshold schemes over any finite Abelian group. In *SIAM Journal on Discrete Mathematics*, 7(4):667-679, 1994.
20. M. Fitzi, M. Hirt, and U. Maurer. Trading correctness for privacy in unconditional multi-party computation. In *Proc. of CRYPTO '98*, LNCS 1462, pp. 121-136, 1998.
21. U. Feige, J. Kilian, and M. Naor. A minimal model for secure computation. In *Proc. of 26th STOC*, pp. 554-563, 1994.
22. M. Fitzi and U. Maurer Efficient Byzantine agreement secure against general adversaries. In *Proc. of DISC '98*, LNCS 1499, pp. 134-148, 1998.

23. S. Fehr and U. Maurer Linear VSS and distributed commitments based on secret sharing and pairwise checks. In *Proc. of CRYPTO '02*, LNCS 2442, pp. 565-580.
24. M. Franklin and M. Yung. Communication complexity of secure computation. In *Proc. of 24th STOC*, pp. 699-710, 1992.
25. R. Gennaro, M. O. Rabin, and T. Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *Proc. of 17th PODC*, pp. 101-111, 1998.
26. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game (extended abstract). In *Proc. of 19th STOC*, pp. 218-229, 1987.
27. M. Hirt and U. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In *Proc. of 16th PODC*, 1997, pp. 25-34.
28. M. Hirt and U. Maurer. Robustness for free in unconditional multi-party computation. In *Proc. of CRYPTO '01*, LNCS 2139, pp. 101-118, 2001.
29. M. Hirt, U. Maurer, and B. Przydatek. Efficient secure multi-party computation. In *Proc. of ASIACRYPT '00*, LNCS 1976, pp. 143-161, 2000.
30. Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Proc. of 41th FOCS*, pp. 294-304, 2000.
31. Y. Ishai and E. Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *Proc. of 29th ICALP*, pp. 244-256, 2002.
32. M. Karchmer and A. Wigderson. On span programs. In *Proc. of 8th Conference on Structure in Complexity Theory*, pp. 102-111, 1993.
33. J. Kilian. Founding cryptography on oblivious transfer. In *Proc. of 20th STOC*, pp. 20-31, 1988.
34. Y. Lindell. Parallel coin-tossing and constant-round secure two-party Computation. In *Proc. of CRYPTO '01*, LNCS 2139, pp. 171-189, 2001.
35. U. Maurer. Secure multi-party computation made simple. In *Proc. of SCN '02*, LNCS 2576, pp. 14-28, 2002.
36. M. Naor, and K. Nissim. Communication Preserving Protocols for Secure Function Evaluation. In *Proc. of 33rd STOC*, pp. 590-599, 2001.
37. R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *Proc. of 10th PODC*, pp. 51-59, 1991.
38. T. Sandler, A. Young, and M. Yung. Non-interactive cryptocomputing for NC^1. In *Proc. of 40th FOCS*, pp. 554-567, 1999.
39. A. Shamir. How to share a secret. *CACM*, 22(11):612-613, 1979.
40. A. C. Yao. Protocols for secure computations. In *Proc. of 23th FOCS*, pp. 160-164, 1982.
41. A. C. Yao. How to generate and exchange secrets. In *Proc. of 27th FOCS*, pp. 162-167, 1986.

A Proof of Lemma 1

Proof of Lemma 1: Write $\alpha = \overline{X} \in \Lambda$ (the residue class of X modulo $f(X)$). Then for each $\lambda \in \Lambda$, there exists a unique coordinate vector $\overrightarrow{\lambda} = (\lambda_0, \dots, \lambda_{m-1})^T \in \mathbb{Z}^m$ such that $\lambda = \lambda_0 + \lambda_1 \cdot \alpha + \cdots + \lambda_{m-1} \cdot \alpha^{m-1}$. In other words, $\{1, \alpha, \dots, \alpha^{m-1}\}$ is a basis for Λ when viewed as a $\mathbb{Z}$-module. For each $\mu \in \Lambda$ there exists a matrix in $\mathbb{Z}^{m \times m}$, denoted as $[\mu]$, such that $[\mu]\overrightarrow{\lambda} = \overrightarrow{\mu\lambda}$ (the coordinate vector of $\mu\lambda$) for every $\lambda \in \Lambda$. The columns of $[\mu]$ are simply the coordinate vectors of $\mu, \mu \cdot \alpha, \dots, \mu \cdot \alpha^{m-1}$. If $\mu \in \mathbb{Z}$, then $[\mu]$ is a diagonal matrix with μ's on the diagonal. Furthermore, for all $\lambda, \mu \in \Lambda$, we have the identities $[\lambda + \mu] = [\lambda] + [\mu]$ and $[\lambda\mu] = [\lambda][\mu]$.

Consider a span program $\mathcal{M} = (\Lambda, M, \psi, \boldsymbol{\varepsilon})$ over Λ for an adversary structure $\mathcal{A}$. Write d (resp. e) for the number of rows (resp. columns) of M. First, we define an integer span program $\tilde{\mathcal{M}} = (\mathbb{Z}, \tilde{M}, \bar{\psi}, \tilde{\boldsymbol{\varepsilon}})$ as follows. Construct $\tilde{M} \in \mathbb{Z}^{md \times me}$ from M by replacing each entry $\mu \in \Lambda$ in M by the matrix $[\mu]$. The labeling ψ is extended to $\bar{\psi}$ in the obvious way, i.e., if a row in M is labelled by a player P_i, then the rows that it is substituted with in $\tilde{M}$ are labelled by the same player P_i. The corresponding target vector is defined by $\tilde{\boldsymbol{\varepsilon}} = (1, 0 \dots, 0)^T \in \mathbb{Z}^{me}$.

We verify that $\tilde{\mathcal{M}}$ is a span program for the same adversary structure $\mathcal{A}$ (however, it is not necessarily (strongly) multiplicative, even if $\mathcal{M}$ is). First, consider a set $A \notin \mathcal{A}$. By definition, there exists a vector $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_{d_A})^T \in \Lambda^{d_A}$ such that $M_A^T \boldsymbol{\lambda} = \boldsymbol{\varepsilon}$, respectively $\boldsymbol{\lambda}^T M_A = \boldsymbol{\varepsilon}^T$, where d_A denotes the number of rows of M_A. Using the identities stated above and carrying out the matrix multiplication "block-wise", it follows that $([\lambda_1], \ldots, [\lambda_{d_A}])\tilde{M}_A = ([1], [0], \ldots, [0])$. Hence, if $\bar{\boldsymbol{\lambda}}^T \in \mathbb{Z}^{md_A}$ denotes the first row of the matrix $([\lambda_1], \ldots, [\lambda_{d_A}])$ then $\bar{\boldsymbol{\lambda}}^T \tilde{M}_A = \tilde{\boldsymbol{\varepsilon}}^T$, respectively $\tilde{M}_A^T \bar{\boldsymbol{\lambda}} = \tilde{\boldsymbol{\varepsilon}}$. Now, consider a set $A \in \mathcal{A}$. Hence, there exists $\boldsymbol{\kappa} = (\kappa_1, \kappa_2, \ldots, \kappa_e)^T \in \Lambda^e$ such that $\kappa_1 = 1$ and $M_A \boldsymbol{\kappa} = \mathbf{0} \in \Lambda^{d_A}$. Using similar reasoning as above, it follows that

$$\tilde{M}_A \begin{pmatrix} [\kappa_1] \\ \vdots \\ [\kappa_e] \end{pmatrix} = \begin{pmatrix} [0] \\ \vdots \\ [0] \end{pmatrix}$$

Hence, if $\tilde{\boldsymbol{\kappa}} \in \mathbb{Z}^{me}$ denotes the first column of the matrix derived from $\boldsymbol{\kappa}$ in the above equation, then the first m entries of $\tilde{\boldsymbol{\kappa}}$ are $1, 0, \ldots, 0$ (since $\kappa_1 = 1$) and $\tilde{M}_A \tilde{\boldsymbol{\kappa}} = \mathbf{0}$.
The claimed span program $\bar{\mathcal{M}} = (\Lambda, \bar{M}, \bar{\psi}, \bar{\boldsymbol{\varepsilon}})$ is now constructed from $\tilde{\mathcal{M}}$ by deleting the 2nd up to m-th leftmost columns of $\tilde{M}$ and the corresponding coordinates of $\tilde{\boldsymbol{\varepsilon}}$. $\bar{\mathcal{M}}$ still accepts the sets $A \notin \mathcal{A}$ and, by the special form of the vector $\tilde{\boldsymbol{\kappa}}$ above, rejects every set $A \in \mathcal{A}$.

Concerning multiplication, note first that for any fixed $\delta \in \Lambda$ there exists a matrix in $\Lambda^{m \times m}$, denoted as $[\![\delta]\!]$, such that for all $\lambda, \mu \in \Lambda$ the first coordinate of (the coordinate vector of) $\lambda\delta\mu$ equals $\overrightarrow{\lambda}^T [\![\delta]\!] \overrightarrow{\mu}$. The (i,j)-th entry of $[\![\delta]\!]$ is given by the first coordinate of (the coordinate vector of) $\delta\alpha^{i-1}\alpha^{j-1}$, as can easily be verified. It is also easy to verify that $[\lambda]^T [\![\delta]\!] [\mu] = [\![\lambda\delta\mu]\!]$.
Let now $D \in \Lambda^{d \times d}$ be a block-diagonal matrix as required in Definition 3 such that $M^T D M = \boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T$. Recall that $\tilde{M}$ has been constructed from M by replacing every entry μ by $[\mu]$. Construct $\bar{D} \in \mathbb{Z}^{md \times md}$ from D by replacing every entry δ of D by $[\![\delta]\!]$. It follows that $\tilde{M}^T \bar{D} \tilde{M} = E'$, where E' has the matrix $[\![1]\!]$ in the upper left corner and zeros elsewhere, and where $[\![1]\!]$ itself has a 1 in the upper left courner. Therefore, the matrix $\bar{M}$, which is constructed from $\tilde{M}$ by removing the 2nd up to the m-th first columns, fulfills $\bar{M}^T \bar{D} \bar{M} = \bar{\boldsymbol{\varepsilon}}\bar{\boldsymbol{\varepsilon}}^T$, as required.
The corresponding claim concerning the *strong* multiplication property follows by applying the above to M_{A^c}, where A^c is the complement $A^c = \{1, \ldots, n\} \setminus A$ of a set $A \in \mathcal{A}$. □

B Building Blocks for *Actively* Secure MPC

In this section, we present the protocols discussed in Section 3.4 that allow the MPC protocol to withstand *active* attacks: a linear distributed commitment, and the auxiliary protocols CTP, CSP and CMP. They are straight-forward generalizations of the field-based protocols presented in [14].

Throughout the section, let Λ be a commutative ring with 1, and let $\mathcal{M} = (\Lambda, M, \psi, \boldsymbol{\varepsilon})$ be a span program for a Q^3 adversary structure $\mathcal{A}$. For the perfectly secure CMP, we additionally require $\mathcal{M}$ to be strongly multiplicative. Finally, let R be an arbitrary finite Λ-algebra. The following protocols are secure with respect to an adversary corrupting the players of an arbitrary set $A \in \mathcal{A}$.

Linear Distributed Commitment. A commitment C of a secret $s \in R$ is defined to be a *correct* sharing $\mathbf{s} = M\mathbf{b}$ of s, distributed among the players. To *open* such a commitment C, the commiter reveals the corresponding sharing vector $\mathbf{b}$ and every player reveals his share $\mathbf{s}_i$, and the opening is accepted if and only if $M_i\mathbf{b} = \mathbf{s}_i$ for all players P_i, except those of a set $A \in \mathcal{A}$. The hiding property follows from the privacy of the secret sharing scheme, and the binding property can be verified using the Q^3 property of $\mathcal{A}$. To *commit* to a secret s, the commiter has to share s in such a way that the sharing is guaranteed to be correct. This can be achieved as follows. Instead of the vector $\mathbf{b}$, the dealer/commiter chooses a random symmetrical matrix B with the secret s in the upper left corner and sends $U_i = M_i B$ to player P_i. By pairwise checking, public complaining and accusing it is then enforced that $M_j U_i^T = U_j M_i^T$ for every pair P_i, P_j of honest players. This then implies that there exists a vector $\mathbf{b}$ such that $\mathbf{s}_i = M_i\mathbf{b}$ for every honest player P_i, where $\mathbf{s}_i$ denotes the first column of U_i, meaning that the honest players hold a correct sharing of some secret s. Indeed, if A collects the honest players, then $M_A U_A^T = U_A M_A^T$ implies that

$$\mathbf{s}_A = U_A \boldsymbol{\varepsilon} = U_A M_A^T \boldsymbol{\lambda} = M_A U_A^T \boldsymbol{\lambda} = M_A \mathbf{b}$$

for $\boldsymbol{\lambda}$ such that $M_A^T \boldsymbol{\lambda} = \boldsymbol{\varepsilon}$, and for $\mathbf{b} = U_A^T \boldsymbol{\lambda}$.

Commitment Transfer Protocol. The purpose of a CTP is to allow a player P_j to transfer a commitment C of a secret s, which in our case is a correct sharing $\mathbf{s} = M\mathbf{b}$ of s, to another player P_k. It must be guaranteed that this protocol leaks no information to the adversary if P_j and P_k are honest, but also that the new commitment contains the same value as the old, even if P_j and P_k are both corrupt.

If, for every set $A \notin \mathcal{A}$, the span program fulfills the stronger condition $\mathrm{im}(M_A^T) = \Lambda^e$ (rather than $\boldsymbol{\varepsilon} \in \mathrm{im}(M_A^T)$)[22], then the CTP works as simple as with cryptographic commitments. Namely, to transfer a commitment C to P_k, C is opened privately to P_k. In our case, this means that P_j sends $\mathbf{b}$ and every P_i sends $\mathbf{s}_i$ privately to P_k, and P_k accepts the opening if (and only if) $\mathbf{s}_i = M_i\mathbf{b}$ holds except for players P_i of a possibly corrupted player set $A \in \mathcal{A}$. P_k's commitment for s is now the same sharing $\mathbf{s}$, and $\mathbf{b}$ is the corresponding "opening information".

However, in the general case, this would *not* lead to a secure CTP, as the adversary could achieve that an honest player P_k cannot correctly open the commitment that has been transfered to him by a corrupted player P_j, making him look like a cheater. This can be overcome as follows (see also [14]). P_k does not adopt P_j's commitment C but he generates a new commitment C' for s by the commit protocol, and he opens the difference $C' - C$ to zero (and if he does not succeed, P_j has to open C' in public).

Commitment Sharing Protocol. A CSP allows a dealer, who is committed to a secret s, to share s such that every player will be committed to his share, and, on the other hand, it is guaranteed that indeed s is correctly shared. A CSP can generically be constructed from the linear distributed commitment and the corresponding CTP. Namely, the dealer chooses a random sharing vector $\mathbf{b}$ with s as first entry and commits to the random entries (he is already committed to s), and he transfers the resulting commitments for the shares $\mathbf{s} = M\mathbf{b}$ to the corresponding players, using the CTP. Clearly, this works similarly if the sharing should be with respect to a different span program matrix M^*.

Commitment Multiplication Protocol. A CMP allows a player, who is committed to secrets s, s' and to the product $s'' = ss'$, to prove that indeed $s'' = ss'$. The first solution is based on an ordinary span program, but is "only" unconditionally secure, while the second is perfectly secure but requires a *strongly multiplicative* span program.

Unconditional CMP: Let C, C' and C'' be the commitments of s, s' and $s'' = ss'$, respectively. In order to prove that $s'' = ss'$, the prover chooses a random $d \in R$, and commits to d and ds'. Let D and E denote the resulting commitments. The players jointly generate a random challenge $c \in \{0, 1\}$ using standard techniques. Then, the prover opens the commitment $c \cdot C + D$ to $r = cs' + d$ and the commitment $r \cdot C' - E - c \cdot C''$ to 0. The proof is accepted if both openings are accepted.

It is easy to see that the opened values give no extra information to the adversary, while the proof is rejected if $s'' \neq ss'$ except with probability 1/2, which can be made negligibly small by repetition.

Perfect CMP: First,we introduce some new notation. Let $\mathbf{x} = (x_1, \ldots, x_m)$ and $\mathbf{y} = (y_1, \ldots, y_m)$ be two row (or column) vectors over an arbitrary ring R. Then $\mathbf{x} \otimes \mathbf{y}$ denotes the row (or column) vector consisting of all products $x_i y_j$:

$$\mathbf{x} \otimes \mathbf{y} = (x_1y_1, \ldots, x_1y_m; x_2y_1, \ldots, x_2y_m; \ldots\ldots; x_my_1 \ldots, x_my_m)$$

[22] This e.g. holds for threshold span programs resulting from Proposition 1.

Let additionally $\varphi : \{1, \ldots, m\} \to \mathcal{P}$ be a labelling function. Then $\mathbf{x} \otimes_\varphi \mathbf{y}$ denotes the row (or column) vector consisting of all products $x_i y_j$ with $\varphi(i) = \varphi(j)$:

$$\mathbf{x} \otimes_\varphi \mathbf{y} = (\mathbf{x}_1 \otimes \mathbf{y}_1, \ldots, \mathbf{x}_n \otimes \mathbf{y}_n)$$

Now, the perfectly secure CMP is based on the fact that if $\mathcal{M} = (\varLambda, M, \psi, \boldsymbol{\varepsilon})$ is strongly multiplicative, then there exists a span program $\mathcal{M}^*$ such that

1. if $\mathbf{s}$ and $\mathbf{s}'$ are sharings of s and s' with respect to $\mathcal{M}$, then $\mathbf{s}'' = \mathbf{s} \otimes_\psi \mathbf{s}'$ is a sharing of $s'' = ss'$ with respect to $\mathcal{M}^*$, and
2. if a set A is rejected by $\mathcal{M}$, then $A^c = \mathcal{P} \setminus A$ is accepted by $\mathcal{M}^*$.

For instance, in the threshold case, when s and s' have been shared using two polynomials of degree t, multiplying corresponding shares of s and s' yields a sharing of ss', but with respect to a polynomial of degree $2t$. In general, $\mathcal{M}^* = (\varLambda, M^*, \psi^*, \boldsymbol{\varepsilon}^*)$ is constructed as follows. For every $P_i \in \mathcal{P}$, and for every pair of (not necessarily different) rows $\mathbf{m}$ and $\mathbf{m}'$ of M that are labeled by P_i, let $\mathbf{m} \otimes \mathbf{m}'$ be a row of M^*, labeled by P_i. It is not hard to verify that condition 1. follows by construction and 2. from the strong multiplication property.

The CMP now works as follows. The prover shares s and s' with respect to M using the CSP, resulting in sharings $\mathbf{s} = M\mathbf{b}$ and $\mathbf{s}' = M\mathbf{b}'$, respectively, and commitments for all shares. Furthermore, again using CSP, he shares $s'' = ss'$ with respect to M^*, resulting in a sharing $\mathbf{s}'' = M^*\mathbf{b}''$ and corresponding commitments. However, he chooses $\mathbf{b}''$ in such a way that $\mathbf{s}'' = \mathbf{s} \otimes_\psi \mathbf{s}'$. Every player P_i now verifies whether indeed $\mathbf{s}''_i = \mathbf{s}_i \otimes \mathbf{s}'_i$, and, in case this does not hold, shows the dishonestness of the dealer by opening the commitments for $\mathbf{s}_i$, $\mathbf{s}'_i$ and $\mathbf{s}''_i$. If no such triple is opened, the proof is accepted. Therefore, if the proof is accepted, then $\mathbf{s}''_{A^c} = \mathbf{s}_{A^c} \otimes_\psi \mathbf{s}'_{A^c}$, where $A \in \mathcal{A}$ collects the corrupted players, and since A^c is accepted by $\mathcal{M}^*$, the secret "behind" $\mathbf{s}''_{A^c}$ is uniquely defined and thus equal to ss'.

C Alternative Protocols for the Maximum Function

In this section we survey some alternatives to the protocol for the maximum function from Section 5, none of which matches both its round complexity and its communication complexity.

We start by noting that with a higher round complexity it is possible to obtain much better asymptotic communication complexity. In fact, using techniques from [36] it is possible to obtain a protocol with a poly-logarithmic number of rounds and with communication complexity as low as $\mathrm{poly}(n, \log\log M)$. Settling for $\mathrm{poly}(n, \log M)$ communication, it is possible to obtain protocols whose round complexity corresponds to that of degree-3 polynomials [30, 31]. Using a careful implementation of the protocol from [5], it is possible to obtain a *computationally* secure protocol with the same round complexity (corresponding to degree-3 polynomials) and with somewhat better asymptotic communication complexity (logarithmic in M instead of poly-logarithmic). However, our goal here is to use even fewer rounds by reducing the secure evaluation of the maximum function to that of degree-2 polynomials.

To this end, let X denote an $M \times n$ matrix over $\mathbb{F}_Q$, where the ith column of X is "owned" by player P_i. Consider the following degree-2 randomizing polynomials vector over $\mathbb{F}_Q$:

$$p(X, r) = Xr,$$

where r is a column vector of n random inputs. If each player P_i picks the ith column of X so that its first y_i entries are 1 and the remaining entries are 0, the output distribution $P(X)$ will contain random field elements in its first $\max(y_1, \ldots, y_n)$ entries, and 0 in the remaining entries. This apparently gives a satisfactory solution to the problem. Unfortunately, the security of the corresponding protocol can be broken by malicious players who do not follow the above procedure for picking their columns of

X. For instance, suppose that P_1 lets the first column of X be $(0, 1, 0, 0, \ldots, 0)$. If all the remaining inputs are 0, the output will reveal that (with high probability) some player has cheated. However, using this strategy, P_1 is still able to simultaneously achieve the following: (1) learn whether all of the remaining inputs are 0; (2) set the maximum to 2 in case that the maximum of the remaining inputs is 1. Note that each of (1) and (2) can be *separately* achieved in an ideal evaluation of the maximum function (by picking the input to be 0 or 2, respectively), but it is impossible to achieve both simultaneously.

The above security flaw can be fixed as follows. Let $f(X)$ be a function returning the index of the bottom-most non-zero row of X. The secure evaluation of the maximum function reduces to the secure evaluation of $f(X)$, where again player P_i is free to determine the ith column of X. Now, f can be represented by the degree-2 polynomial vector $p = (p_0, ..., p_M)$, where p_j is a random linear combination of the $(j + 1)n$ entries in the last $j + 1$ rows of X, and each p_j uses a disjoint set of random inputs. This gives a fully secure solution to the maximum function with the desired round complexity. However, the number of random inputs in the above representation is quadratic in M, resulting in quadratic communication complexity.

To summarize, all the alternative solutions we have presented either require more rounds, require a higher communication complexity, or fail to be secure against an active adversary.