Secure Arithmetic Computation with No Honest Majority

Yuval Ishai * Manoj Prabhakaran † Amit Sahai ‡

November 8, 2008

Abstract

We study the complexity of securely evaluating arithmetic circuits over finite rings. This question is motivated by natural secure computation tasks. Focusing mainly on the case of *two-party* protocols with security against *malicious* parties, our main goals are to: (1) only make black-box calls to the ring operations and standard cryptographic primitives, and (2) minimize the number of such black-box calls as well as the communication overhead.

We present several solutions which differ in their efficiency, generality, and underlying intractability assumptions. These include:

- An *unconditionally secure* protocol in the OT-hybrid model which makes a black-box use of an arbitrary ring R, but where the number of ring operations grows linearly with (an upper bound on) $\log |R|$.
- Computationally secure protocols in the OT-hybrid model which make a black-box use of an underlying ring, and in which the number of ring operations does not grow with the ring size. The protocols rely on variants of previous intractability assumptions related to linear codes. In the most efficient instance of these protocols, applied to a suitable class of fields, the (amortized) communication cost is a constant number of field elements per multiplication gate and the computational cost is dominated by $O(\log k)$ field operations per gate, where k is a security parameter. These results extend a previous approach of Naor and Pinkas for secure polynomial evaluation (*SIAM J. Comput.*, 35(5), 2006).
- A protocol for the rings $\mathbb{Z}_m = \mathbb{Z}/m\mathbb{Z}$ which only makes a black-box use of a homomorphic encryption scheme. When m is prime, the (amortized) number of calls to the encryption scheme for each gate of the circuit is constant.

All of our protocols are in fact *UC-secure* in the OT-hybrid model and can be generalized to *multiparty* computation with an arbitrary number of malicious parties.

*Technion, Israel and University of California, Los Angeles. yuvali@cs.technion.il

†University of Illinois, Urbana-Champaign. mmp@cs.uiuc.edu

‡University of California, Los Angeles. sahai@cs.ucla.edu

1 Introduction

This paper studies the complexity of secure multiparty computation (MPC) tasks which involve *arithmetic* computations. Following the general feasibility results from the 1980s [Yao86, GMW87, BGW88, CCD88], much research in this area shifted to efficiency questions, with a major focus on the efficiency of securely distributing natural computational tasks that arise in the "real world". In many of these cases, some inputs, outputs, or intermediate values in the computation are integers, finite-precision reals, matrices, or elements of a big finite ring, and the computation involves arithmetic operations in this ring. To name just a few examples from the MPC literature, such arithmetic computations are useful in the contexts of distributed generation of cryptographic keys [BF01, FMY98, PS98, Gil99, ACS02], privacy-preserving data-mining and statistics [LP02, CIK$^+$01], comparing and matching data [NP06, FNP04, HL08], auctions and mechanism design [NPS99, DFK$^+$06, Tof07, BCD$^+$08], and distributed linear algebra computations [CD01, NW06, KMWF07, CKP07, MW08].

This motivates the following question:

> What is the complexity of securely evaluating a given arithmetic circuit C over a given finite ring R?

Before surveying the state of the art, some clarifications are in place.

Arithmetic circuits. An arithmetic circuit over a ring is defined similarly to a standard boolean circuit, except that the inputs and outputs are ring elements rather than bits and gates are labeled by the ring operations add, subtract, and multiply. (Here and in the following, by "ring" we will refer to a finite ring by default.) In the current context of distributed computations, the inputs and outputs of the circuit are annotated with the parties to which they belong. Thus, the circuit C together with the ring R naturally define a multi-party arithmetic functionality C^R. Note that arithmetic computations over the integers or finite-precision reals can be embedded into a sufficiently large finite ring or field, provided that there is an a-priori upper bound on the bit-length of the output. See Section 1.4 for further discussion of the usefulness of arithmetic circuits and some extensions of this basic model to which our results apply.

Secure computation model. The main focus of this paper is on secure *two-party* computation or, more generally, MPC with an arbitrary number of malicious parties. (In this setting it is generally impossible to guarantee output delivery or even fairness, and one has to settle for allowing the adversary to abort the protocol after learning the output.) Our protocols are described in the "OT-hybrid model," namely in a model that allows parties to invoke an ideal oblivious transfer (OT) oracle [Rab81, EGL85, Gol04]. This has several advantages in generality and efficiency, see [IPS08] and Section 1.4 below for discussion.

Ruling out the obvious. An obvious approach for securely realizing an arithmetic computation C^R is by first designing an equivalent *boolean* circuit C' which computes the same function on a binary representation of the inputs, and then using standard MPC protocols for realizing C'. The main disadvantage of such an approach is that it typically becomes very inefficient when R is large. One way to rule out such an approach, at least given the current state of the art, is to require the communication complexity to grow at most linearly with $\log |R|$. (Note that even in the case of finite fields with n-bit elements, the size of the best known boolean multiplication circuits is $\omega(n \log n)$; the situation is significantly worse for other useful rings, such as matrix rings.)

A cleaner way for ruling out such an approach, which is of independent theoretical interest, is by restricting protocols to only make a *black-box* access to the ring R. That is, Π securely realizes C if Π^R securely

realizes C^R for every finite ring R and *every representation of elements in* R.[1] This black-box access to R enables Π to perform ring operations and sample random ring elements, but the correspondence between ring elements and their identifiers (or even the exact size of the ring) will be unknown to the protocol.[2] When considering the special case of fields, we allow by default the protocol Π to access an inversion oracle.

1.1 Previous Work

In the setting of MPC *with honest majority*, most protocols from the literature can make a black-box use of an arbitrary *field*. An extension to arbitrary black-box rings was given in [CFIK03], building on previous black-box secret sharing techniques of [DF89, CF02].

In the case of secure two-party computation and MPC with no honest majority, most protocols from the literature apply to boolean circuits. Below we survey some previous approaches from the literature that apply to secure arithmetic computation with no honest majority.

In the semi-honest model, it is easy to employ any homomorphic encryption scheme with plaintext group $\mathbb{Z}_m$ for performing arithmetic MPC over $\mathbb{Z}_m$. (See, e.g., [AF90, CIK$^+$01].) An alternative approach, which relies on oblivious transfer and uses the standard binary representation of elements in $\mathbb{Z}_m$, was employed in [Gil99]. These protocols make a black-box use of the underlying cryptographic primitives but do not make a black-box use of the underlying ring. Applying the general compilers of [GMW87, CLOS02] to these protocols in order to obtain security in the malicious model would result in inefficient protocols which make a non-black-box use of the underlying cryptographic primitives (let alone the ring).

In the *malicious model*, protocols for secure arithmetic computation based on *threshold* homomorphic encryption were given in [CDN01, DN03][3] (extending a similar protocol for the semi-honest model from [FH96]). These protocols provide the most practical general solutions for secure arithmetic two-party computation we are aware of, requiring a constant number of modular exponentiations for each arithmetic gate. On the down side, these protocols require a nontrivial setup of keys which is expensive to distribute. Moreover, similarly to all protocols described so far, they rely on special-purpose zero-knowledge proofs and specific number-theoretic assumptions and thus do not make a black-box use of the underlying cryptographic primitives, let alone a black-box use of the ring.

The only previous approach which makes a black-box use of an underlying ring (as well as a black-box use of OT) was suggested by Naor and Pinkas [NP06] in the context of secure polynomial evaluation. Their protocol can make a black-box use of any *field* (assuming an inversion oracle), and its security is related to the conjectured intractability of decoding Reed-Solomon codes with a sufficiently high level of random noise. The protocol from [NP06] can be easily used to obtain general secure protocols for arithmetic circuits in the *semi-honest* model. However, extending it to allow full simulation-based security in the malicious model (while still making only a black-box use of the underlying field) is not straightforward. (Even in the special case of secure polynomial evaluation, an extension to the malicious model suggested in [NP06] only considers *privacy* rather than full simulation-based security.)

Finally, we note that Yao's garbled circuit technique [Yao86], which is essentially the only known technique for constant-round secure computation of general functionalities, does not have a known arithmetic

[1] When considering computational security we will require representations to be *computationally efficient*, in the sense that given identifiers of two ring elements a, b one can efficiently compute the identifiers of $a + b$, $a - b$, and $a \cdot b$.

[2] Note that it is not known how to efficiently learn the structure of a ring using a black box access to ring operations, even in the special case of general finite fields [BL96, MR07].

[3] While [CDN01, DN03] refer to the case of robust MPC in the presence of an honest majority, these protocols can be easily modified to apply to the case of MPC with no honest majority. We note that while a main goal of these works was to minimize the growth of complexity with the number of parties, we focus on minimizing the complexity in the two-party case.

analogue. Thus, in all general-purpose protocols for secure arithmetic computation (including the ones presented in this work) the round complexity must grow with the multiplicative depth[4] of C.

1.2 Our Contribution

We study the complexity of general secure arithmetic computation over finite rings in the presence of an arbitrary number of malicious parties. We are motivated by the following two related goals.

- *Black-box feasibility*: only make a black-box use of an underlying ring R or field F and standard cryptographic primitives;
- *Efficiency*: minimize the number of such black-box calls, as well as the communication overhead.

For simplicity, we do not attempt to optimize the dependence of the complexity on the number of parties, and restrict the following discussion to the two-party case.

We present several solutions which differ in their efficiency, generality, and underlying intractability assumptions. Below we describe the main protocols along with their efficiency and security features. An overview of the underlying techniques is presented in Section 1.3.

An unconditionally secure protocol. We present an *unconditionally secure* protocol in the OT-hybrid model which makes a *black-box* use of an *arbitrary* finite ring R, but where the number of ring operations and the number of ring elements being communicated grow linearly with (an upper bound on) $\log |R|$. (We assume for simplicity that an upper bound on $\log |R|$ is given by the ring oracle, though such an upper bound can always be inferred from the length of the strings representing ring elements.) More concretely, the number of ring operations for each gate of C is $poly(k) \cdot \log |R|$, where k is a statistical security parameter. This gives a two-party analogue for the MPC protocol over black-box rings from [CFIK03], which requires an honest majority (but does not require the number of ring operations to grow with $\log |R|$).

Protocols based on noisy linear encodings. Motivated by the goal of reducing the overhead of the previous protocol, we present a general approach for deriving secure arithmetic computation protocols over a ring R from linear codes over R. The (computational) security of the protocols relies on intractability assumptions related to the hardness of decoding in the presence of random noise. These protocols generalize and extend in several ways the previous approach of Naor and Pinkas for secure polynomial evaluation [NP06] (see Section 1.3 for discussion). Using this approach, we obtain the following types of protocols in the OT-hybrid model.

- A protocol which makes a black-box use of an arbitrary *field* F, in which the number of field operations (and field elements being communicated) does not grow with the field size. More concretely, the number of field operations for each gate of C is bounded by a fixed polynomial in the security parameter k, independently of $|F|$. The underlying assumption is related to the conjectured intractability of decoding a random linear code[5] over F. Our assumption is implied by the assumption that a noisy codeword in a random linear code over F is pseudorandom. Such a pseudorandomness assumption follows from the average-case hardness of *decoding* a random linear code when the field size is polynomial in k (see [BFKL93, AIK07] for corresponding reductions in the binary case).

[4] The *multiplicative depth* of a circuit is the maximal number of multiplication gates on a path from an input to an output.

[5] The above efficiency feature requires that random linear codes remain hard to decode even over very large fields. Note, however, that $\log |F|$ is effectively restricted by the running time of the adversary, which is (an arbitrarily large) polynomial in k. The assumption can be relaxed if one allows the number of ring operation to moderately grow with $\log |F|$.

- A variant of the previous protocol which makes a black-box use of an arbitrary *ring* R, and in particular does not rely on inversion. This variant is based on families of linear codes over rings in which decoding in the presence of erasures can be done efficiently, and for which decoding in the presence of (a suitable distribution of) random noise seems intractable.

- The most efficient protocol we present relies on the intractability of decoding Reed-Solomon codes with a (small) constant rate in the presence of a (large) constant fraction of noise.[6] The amortized communication cost is a constant number of field elements per multiplication gate. (Here and in the following, when we refer to "amortized" complexity we ignore an additive term that may depend polynomially on the security parameter and the circuit depth, but not on the circuit size. In most natural instances of large circuits this additive term does not form an efficiency bottleneck.)

  A careful implementation yields protocols whose amortized computational cost is $O(\log k)$ field operations per gate, where k is a security parameter, assuming that the field size is super-polynomial in k. In contrast, protocols which are based on homomorphic encryption schemes (such as [CDN01] or the ones obtained in this work) apply modular exponentiations, which require $\Omega(k + \log |F|)$ ring multiplications per gate, in a ciphertext ring which is larger than F. This is the case even in the semi-honest model. Compared to the "constant-overhead" protocol from [IKOS08] (applied to a boolean circuit realizing C^F), our protocol has better communication complexity and relies on a better studied assumption, but its asymptotic computational complexity is worse by an $O(\log k)$ factor when implemented in the *boolean* circuit model.

Protocols making a black-box use of homomorphic encryption. For the case of rings of the form $\mathbb{Z}_m = \mathbb{Z}/m\mathbb{Z}$ (with the standard representation) we present a protocol which makes a black-box use of any homomorphic encryption scheme with plaintext group $\mathbb{Z}_m$. Alternatively, the protocol can make a black-box use of homomorphic encryption schemes in which the plaintext group is determined by the key generation algorithm, such as those of Paillier [Pai99] or Damgård-Jurik [DJ02]. In both variants of the protocol, the (amortized) number of communicated ciphertexts and calls to the encryption scheme for each gate of C is constant, assuming that m is prime. This efficiency feature is comparable to the protocols from [CDN01, DN03] discussed in Section 1.1 above. Our protocols have the advantages of using a more general primitive and only making a *black-box* use of this primitive (rather than relying on special-purpose zero-knowledge protocols). Furthermore, the additive term which we ignore in the above "amortized" complexity measure seems to be considerably smaller than the cost of distributing the setup of the threshold cryptosystem required by [CDN01].

Both variants of the protocol can be naturally extended to the case of matrix rings $\mathbb{Z}_m^{n \times n}$, increasing the communication complexity by a factor of n^2. (Note that emulating matrix operations via basic arithmetic operations over $\mathbb{Z}_m$ would result in a bigger overhead, corresponding to the complexity of matrix multiplication.) Building on the techniques from [MW08], this protocol can be used to obtain efficient protocols for secure linear algebra which make a black-box use of homomorphic encryption and achieve simulation-based security against malicious parties (improving over similar protocols with security against *covert* adversaries [AL07] recently presented in [MW08]).

All of our protocols are in fact *UC-secure* in the OT-hybrid model and can be generalized to *multiparty* computation with an arbitrary number of malicious parties. The security of the protocols also holds against

[6] The precise intractability assumption we use is similar in flavor to an assumption used in [NP06] for evaluating polynomials of degree $d \geq 2$. With a suitable choice of parameters, our assumption is implied by a natural pseudorandomness variant of the assumption from [NP06], discussed in [KY08]. The assumption does not seem to be affected by the recent progress on list-decoding Reed-Solomon codes and their variants [GS99, CS03, BKY07, PV05].

adaptive adversaries, assuming that honest parties may erase data. (This is weaker than the standard notion of adaptive security [CFGN96] which does not rely on data erasure.) The *round complexity* of all the protocols is a constant multiple of the multiplicative depth of C.

1.3 Techniques

Our results build on a recent technique from [IPS08] (which was inspired by previous ideas from [IKOS07] and also [HIKN08]). The main result of [IPS08] constructs a secure two-party protocol for a functionality f in the OT-hybrid model by making a *black-box* use of the following two ingredients: (1) an *outer MPC protocol* which realizes f using k additional "servers", but only needs to tolerate a *constant fraction* of *malicious* servers; and (2) an *inner two-party protocol* which realizes in the *semi-honest OT-hybrid model* a reactive two-party functionality defined (in a black-box way) by the outer protocol. The latter functionality is essentially a distributed version of the algorithm run by the servers in the outer protocol.

Because of the black-box nature of this construction, if both ingredients make a black-box use of R and/or a black-box use of cryptographic primitives, then so does the final two-party protocol.

Given the above, it remains to find good instantiations for the outer and inner protocols. Fortunately, good instances of the outer protocol already exist in the literature. In the case of general black-box rings, we can use the protocol of [CFIK03]. In the case of fields, we can use a variant of the protocol from [DI06] for better efficiency. This protocol has an amortized communication cost of a constant number of field elements for each multiplication gate in the circuit. In terms of computational overhead, a careful implementation incurs an amortized overhead of $O(\log k)$ field operations per gate, where k is a security parameter, assuming that the field size is superpolynomial in k. (The overhead is dominated by the cost of Reed-Solomon encoding over the field.)

Our final protocols are obtained by combining the above outer protocols with suitable implementations of the inner protocol. Our main technical contribution is in suggesting concrete inner protocols which yield the required security and efficiency features.

Similarly to [IPS08], the inner protocols corresponding to the outer protocols we employ require to securely compute, in the semi-honest model, multiple instances of a simple "product-sharing" functionality, in which Alice holds a ring element a, Bob holds a ring element b, and the output is an additive secret sharing of ab. (The efficient version of the outer protocol requires the inner protocol to perform only a constant amortized number of product-sharings per multiplication gate. All other computations, including ones needed for handling addition gates, are done locally and do not require interaction.) In [IPS08] such a product-sharing protocol is implemented by applying the GMW protocol [GMW87] (in the semi-honest OT-hybrid model) to the binary representation of the inputs. This does not meet our current feasibility and efficiency goals.

Below we sketch the main ideas behind different product-sharing protocols on which we rely, which correspond to the main protocols described in Section 1.2.

Unconditionally secure product-sharing. In our unconditionally secure protocol, Bob breaks his input b into n additive shares and uses them to generate n pairs of ring elements, where in each pair one element is a share of b and the other is a random ring element. (The location of the share of b in each pair is picked at random and is kept secret by Alice. Note that additive secret-sharing can be done using a black-box access to the ring oracle.) Bob sends these n pairs to Alice. Alice multiplies each of the $2n$ ring elements (from the left) by her input a, and subtracts from each element in the i-th pair a random ring element t_i. This results in n new pairs. Bob retrieves from each pair the element corresponding to the original additive share of b by using n invocations of the OT oracle. Bob outputs the sum of the n ring elements she obtained, and Alice

outputs $\sum_{i=1}^{n} t_i$.

It is easy to verify that the protocol has the correct output distribution. The security of the protocol can be analyzed using the Leftover Hash Lemma [ILL89]. (Similar uses of this lemma were previously made in [IN96, IKOS06].) Specifically, the protocol is statistically secure when $n > \log_2 |R| + k$. We note that in light of efficient algorithms for low-density instances of subset sum [LO85], one cannot hope to obtain significant efficiency improvements by choosing a smaller value of n and settling for computational security.

Product-sharing from linear codes. Our construction for black-box fields generalizes the previous approach of Naor and Pinkas [NP06] in a natural way. The high level idea is as follows. Bob sends to Alice a *noisy* randomized linear encoding (or noisy linear secret-sharing) of b which is assumed to hide b. Alice uses the homomorphic properties of this encoding to compute a noisy encoding of $ab + z$ for a random z of her choice. Bob uses OT to retrieve only the non-noisy portions of the latter encoding. Note that the above unconditionally secure protocol can also be viewed as an instance of this general paradigm.

In more detail, suppose that G is an $n \times k$ generating matrix of a linear code $\mathcal{C} \subset F^n$ whose minimal distance is bigger than d. This implies that an encoded message can be efficiently recovered from any $n - d$ coordinates of the encoding by solving a system of linear equations defined by the corresponding sub-matrix of G. Now, suppose that G has the following intractability property: the distribution of $Gu + e$, where u is a random message from F^k whose first coordinate is b and e is a random noise vector of Hamming weight at most d, keeps x semantically secure. (This follows, for instance, from the pseudorandomness of a noisy codeword in the code spanned by all but the first column of G.) Given such G the protocol proceeds as follows. Bob sends to Alice $v = Gu + e$ as above, where e is generated by first picking at random a subset $L \subset [n]$ of size $n - d$ and then picking e_i at random for $i \notin L$ and setting $e_i = 0$ for $i \in L$. By assumption, v keeps b hidden from Alice. Alice now locally computes $v' = a \cdot v - Gz$, where z is a random message in F^k. Restricted to the coordinates in L, this agrees with the encoding of a *random* message whose first coordinate is $ab - z_1$. Using the OT-oracle, Bob obtains from Alice only the coordinates of v' with indices in L, from which it can decode and output $ab - z_1$. Alice outputs z_1.

The basic secure polynomial evaluation protocol from [NP06], when restricted to degree-1 polynomials, essentially coincides with the above protocol when $\mathcal{C}$ is a Reed-Solomon code. The extension to general linear codes makes the underlying security assumption more conservative. Indeed, in contrast to Reed-Solomon codes, the problem of decoding *random* linear codes is believed to be intractable even for very low levels of noise.

In our actual protocols we will use several different distributions for picking the generating matrix G, and allow the noise distribution to depend on the particular choice of G (rather than only on its minimal distance). In particular, for the case of general black-box rings we pick G from a special class of codes for which decoding does not require inversion and yet the corresponding intractability assumption still seems viable.

Finally, in our most efficient code-based protocol we use Reed-Solomon codes as in [NP06], but extend the above general template by letting Bob pack $t = \Omega(k)$ field elements $(b_1, \ldots, b_t)$ into the same codeword v. This variant of the construction does not apply to a general G, and relies on a special property of Reed-Solomon codes which was previously exploited in [FY92]. This approach yields a protocol which realizes t parallel instances of product-sharing by communicating only $O(t)$ field elements.

Product-sharing from homomorphic encryption. Our last product-sharing protocol applies to rings of the form $\mathbb{Z}_m$ or $n \times n$ matrices over such rings and makes a standard use of homomorphic encryption. The only technicality that needs to be addressed is that the most useful homomorphic homomorphic encryption schemes do not allow to control the modulus m but rather have this modulus generated by the key-generation algorithm. However, in the semi-honest model it is simple (via standard techniques) to emulate secure

computation modulo m via secure computation modulo any $M \gg m$.

1.4 Further Discussion

From the OT-hybrid model to the plain model An advantage of presenting our protocols in the OT-hybrid model is that they can be instantiated in a variety of models and under a variety of assumptions. For instance, using UC-secure OT protocols from [PVW08, DNO08], one can obtain efficient UC-secure instances of our protocols in the CRS model. In the stand-alone model, one can implement these OTs by making a black-box use of homomorphic encryption [IKLP06]. Thus, our protocols which make a black-box use of homomorphic encryption do not need to employ an additional OT primitive in the stand-alone model.

We finally note that our protocols requires only $O(k)$ OTs with security in the malicious model, independently of the circuit size; the remaining OT invocations can all be implemented in the semi-honest model, which can be done very efficiently using the technique of [IKNP03]. Furthermore, all the "cryptographic" work for implementing the OTs can be done off-line, before any inputs are available. We expect that in most natural instances of large-scale secure arithmetic computation, the cost of realizing the OTs will not form an efficiency bottleneck.

Extensions. While we explicitly consider here only stateless arithmetic circuits, this model (as well as our results) can be readily generalized to allow stateful, reactive arithmetic computations whose secret state evolves by interacting with the parties.[7]

Another direction for extending the basic results has to do with the richness of the arithmetic computation model. Recall that the standard model of arithmetic circuits allows only to add, subtract, and multiply ring elements. While this provides a clean framework for the study of secure computation over black-box rings, many applications depend on other operations that cannot be efficiently expressed in this basic circuit model. For instance, when emulating arithmetic computation over the integers via computation over a (sufficiently large) finite field, one typically needs to check that the inputs comes from a given range.

As it turns out, reactive arithmetic computations are surprisingly powerful in this context, and can be used to obtain efficient secure realizations of useful "non-arithmetic" manipulations of the state, including decomposing a ring element into its bit-representation, equality testing, inversion, comparison, exponentiation, and others [DFK$^+$06, Tof07]. These reductions enhance the power of the basic arithmetic model, and allow protocols to efficiently switch from one representation to another in computations that involve both boolean and arithmetic operations.

1.5 Roadmap

We now briefly outline the structure of the rest of this paper. Our basic definitions, including those of black-box computational rings and our notion of security in this context, are given in Section 2. To achieve our results (focusing on the two-party setting), recall that our overall technical approach is to invoke [IPS08], which gives a general blueprint for constructing efficient protocols by combining an "outer MPC protocol" secure against active adversaries in the honest majority setting, with an "inner two-party protocol" for simple functionalities that need only be secure against *passive* adversaries. We will give the details of this in Section 5, but the bottom line (as discussed above) is that existing protocols (some with minor modifications) suffice for the outer MPC protocols, and all we need to provide are efficient inner protocols secure against passive

[7] An ideal functionality which formally captures such general reactive arithmetic computations was defined in [DN03] (see also [Tof07, Chapter 4]) and referred to as an *arithmetic black-box* (ABB). All of our protocols for arithmetic circuits can be naturally extended to realize the ABB functionality.

adversaries. Furthermore, since we are in the setting of passive adversaries, the only functionality that we need the inner protocol to compute is a basic ring multiplication function, at the end of which the two parties should hold additive shares of the product of their respective inputs. To construct efficient protocols for this basic functionality, we examine three approaches. Our first two approaches are based on "noisy encodings" of various types, which we define in Section 3, and the last approach is based on homomorphic encryption. The actual protocols ("inner two-party protocols") based on these three approaches are given in Section 4.

2 Preliminaries

Black-box rings and fields. A probabilistic oracle R is said to be a valid implementation of a finite ring R if it behaves as follows: it takes as input one of the commands add, $\mathsf{subtract}$, $\mathsf{multiply}$, sample and two m bit "element identifiers" (or none, in the case of sample), and returns a single m bit string. There is a one-to-one mapping $\mathsf{label} : R \hookrightarrow \{0,1\}^m$ such that for all $x, y \in R$ $\mathsf{R}(\mathsf{op}, \mathsf{label}(x), \mathsf{label}(y)) = \mathsf{label}(x *_R y)$ where op is one of add, $\mathsf{subtract}$ and $\mathsf{multiply}$ and $*_R$ is the ring operation $+, -$, or $\cdot$ respectively. When an input is not from the range of label, the oracle outputs $\perp$. (In a typical protocol, if a $\perp$ is ever encountered by an honest player, the protocol aborts.) The output of $\mathsf{R}(\mathsf{sample})$ is $\mathsf{label}(x)$ where x will be drawn uniformly at random from R. We will be interested in oracles of the kind that implements a *family* of rings, of varying sizes. Such a function should take an additional input id to indicate which ring it is implementing.

Definition 2.1 *A probabilistic oracle $\mathcal{R}$ is said to be a* concrete ring family *(or simply a* ring family*) if, for all strings id, the oracle $\mathcal{R}(\mathsf{id}, \cdot)$ (i.e., with first input being fixed to id), is an implementation of some ring. This concrete ring will be denoted by $\mathcal{R}_{\mathsf{id}}$.*

Note that so far we have not placed any computability requirement on the oracle; we only require a concrete mapping from ring elements to binary strings. However, when considering computationally secure protocols we will typically restrict the attention to "efficient" families of rings: we say $\mathcal{R}$ is a *computationally efficient ring family* if it is a ring family that can be implemented by a probabilistic polynomial time algorithm.

There are some special cases that we shall refer to:

1. Suppose that for all id, we have that $\mathcal{R}_{\mathsf{id}}$ is a ring with an identity for multiplication, 1. Then, we call $\mathcal{R}$ a *ring family with inverse* if in addition to the other operations, $\mathcal{R}(\mathsf{id}, \mathsf{one})$ returns $\mathsf{label}_{\mathsf{id}}(1)$ and $\mathcal{R}(\mathsf{id}, \mathsf{invert}, \mathsf{label}_{\mathsf{id}}(x))$ returns $\mathsf{label}_{\mathsf{id}}(x^{-1})$ if x is a unit (i.e., has a unique left- and right-inverse) and $\perp$ otherwise.

2. If $\mathcal{R}$ is a ring family with inverse such that for all id the ring $\mathcal{R}_{\mathsf{id}}$ is a field, then we say that $\mathcal{R}$ is a *field family*.

3. We call a ring family with inverse $\mathcal{R}$ a *pseudo-field family*, if for all id, all but negligible (in $|\mathsf{id}|$) fraction of the elements in the ring $\mathcal{R}_{\mathsf{id}}$ are units.

Some special families of rings we will be interested in, other than finite fields, include rings of the form $\mathbb{Z}_m = \mathbb{Z}/m\mathbb{Z}$ for a composite integer m (namely, the ring of residue classes modulo m), and rings of matrices over a finite field or ring. With an appropriate choice of parameters, both of these families are in fact pseudo-fields. Note that a concrete ring family $\mathcal{R}$ for the rings of the form $\mathbb{Z}_m$ could use the binary representation of m as the input id; further the elements in $\mathbb{Z}_m$ could be represented as $\lceil \log m \rceil$-bit strings in a natural way. Of course, a different concrete ring family for the same ring can use a different representation.

Finally, for notational convenience we assume that the length of all element identifiers in $\mathcal{R}_{\mathsf{id}}$ is exactly $|\mathsf{id}|$. In particular, the ring $\mathcal{R}_{\mathsf{id}}$ has at most $2^{|\mathsf{id}|}$ elements.

Arithmetic circuits. An arithmetic circuit is a circuit (i.e., a directed acyclic graph with the nodes labeled as input gates, output gates or internal gates), in which the internal gates are labeled with a ring operation: add, subtract or multiply. (In addition, for fields, one often considers the additional constant gate one.) An arithmetic circuit C can be instantiated with any ring R. We denote by C^R the mapping (from a vector of ring elements to a vector of ring elements) defined in a natural way by instantiating C with R. For a concrete ring family $\mathcal{R}$, we denote by $C^{\mathcal{R}}$ the mapping which takes an id and a vector of input identifiers and outputs the corresponding vector of output identifiers. (If any of the inputs is not a valid identifier, $C^{\mathcal{R}}$ outputs $\perp$.)

In the context of multi-party computation, each input or output to such a circuit is annotated to indicate which party (or parties) it "belongs" to. Given such an annotated circuit C and a concrete ring family $\mathcal{R}$, we define the functionality $\mathcal{F}_C^{\mathcal{R}}$ to behave as follows:

- The functionality takes id as a common (public) input, and receives (private) inputs to C from each party. It then evaluates the function $C^{\mathcal{R}}(\mathsf{id}, \text{inputs})$ using access to $\mathcal{R}$, and provides the outputs to the parties.[8]

Protocols securely realizing arithmetic computations. We follow the standard UC-security framework [Can05]. Informally, a protocol π is said to securely realize a functionality $\mathcal{F}$ if there exists a PPT simulator Sim, such that for all (non-uniform PPT) adversaries Adv, and all (non-uniform PPT) environments Env which interact with a set of parties and an adversary, the following two scenarios are indistinguishable: the REAL interaction where the parties run the protocol π and the adversary is Adv; the IDEAL interaction where the parties communicate directly with the ideal functionality $\mathcal{F}$ and the adversary is $\mathsf{Sim}^{\mathsf{Adv}}$. Indistinguishability can either be statistical (in the case of unconditional security) or computational (in the case of computational security). All parties, the adversary, the simulator, the environment and the functionality get the security parameter k as implicit input. Polynomial time computation, computational or statistical indistinguishability and non-uniformity are defined with respect to this security parameter k. However, since we don't impose an a-priori bound on the size of the inputs received from the environment, the running time of honest parties is bounded by a fixed polynomial in the total length of their inputs (rather than a fixed polynomial in k).

We distinguish between *static* corruption and *adaptive* corruption. In the latter case it also makes a difference whether the protocols can erase their state (so that a subsequent corruption will not have access to the erased information), or no erasure is allowed. Our final protocols will have security against adaptive[9] corruption in the model that allows honest parties to erase their state information, but as an intermediate step, we will consider protocols which have security only against static corruption.

We shall consider protocols which make oracle access to a ring family $\mathcal{R}$. For such a protocol we define its *arithmetic computation complexity* as the number of oracle calls to $\mathcal{R}$. Similarly the *arithmetic*

[8] $\mathcal{F}_C^{\mathcal{R}}$ can take id as input from each party, and ensure that all the parties agree on the same id. Alternately, we can restrict to environments which provide the same common input id to all parties. In this case id could be considered part of the specification of the functionality, more appropriately written as $\mathcal{F}_{C,\mathsf{id}}^{\mathcal{R}}$.

[9] One of the reasons for us to aim for adaptive security with erasure is that we will be relying on the main protocol compiler of [IPS08], as described informally in the Introduction and treated more formally in Section 5. This compiler requires that the component protocols, the "outer MPC protocol" and the "inner two-party protocol," both enjoy adaptive security – the outer protocol must be adaptively secure in the model without erasures, but the inner protocol can be adaptively secure with erasures (in the OT-hybrid model). Note that the conference version of [IPS08] incorrectly claimed that the main protocol's proof of security works even when the inner protocol is only statically secure, but this does not seem to be the case. However, this issue does not present any problems for us here, as we are easily able to modify our proposed "inner" protocols to achieve adaptive security with erasures using standard techniques, as detailed in Appendix A.

communication complexity is defined as the number of ring-element labels in the communication transcript. The arithmetic computation (respectively communication) complexity of our protocols will dominate the other computation steps in the protocol execution (respectively, the number of other bits in the transcript). Thus, the arithmetic complexity gives a good measure of efficiency for our protocols.

Note that while any computational implementation of the ring oracle necessarily requires the complexity to grow with the ring size, it is possible that the arithmetic complexity does not depend on the size of the ring at all.

We now define our main notion of secure arithmetic computation.

Definition 2.2 *Let C be an arithmetic circuit. A protocol π is said to be a* secure black-box realization of *C-evaluation for a given set of ring families if, for each $\mathcal{R}$ in the set,*

1. *$\pi^{\mathcal{R}}$ securely realizes $\mathcal{F}_C^{\mathcal{R}}$, and*
2. *the arithmetic (communication and computation) complexity of $\pi^{\mathcal{R}}$ is bounded by some fixed polynomial in k and $|\mathsf{id}|$ (independently of $\mathcal{R}$).*

In the case of unconditional security we will quantify over the set of *all* ring families, whereas in the case of computational security we will typically quantify only over computationally efficient rings or fields.[10] In both cases, the efficiency requirement on π rules out the option of using a brute-force approach to emulate the ring oracle by a boolean circuit.

We remark that our constructions will achieve a stronger notion of security, as the simulator used to establish the security in item (1) above will not depend on $\mathcal{R}$. A bit more precisely, the stronger definition is quantified as follows: there exists a simulator such that for all adversaries, ring families, and environments, the ideal process and the real process are indistinguishable. For simplicity however we phrase our definition as above which does allow different simulators for different $\mathcal{R}$.

3 Noisy Encodings

A central tool for our main protocols is a noisy encoding of elements in a ring or a field. In general this encoding consists of encoding a randomly padded message with a (possibly randomly chosen) linear code, and adding noise to the codeword obtained. The encodings will be such that, with some information regarding the noise, decoding (of a codeword derived from the noisy codeword) is possible, but otherwise the noisy codeword hides the message. The latter will typically be a computational assumption, for parameters of interest to us.

We shall use two kinds of encodings for our basic protocols in Section 4. The first of these encodings has a statistical hiding property which leads to a statistically secure protocol (in the OT-hybrid model). The other kind of encoding we use (described in Section 3.2) is hiding only under computational assumptions. In fact, we provide a general template for such encodings and instantiate it variously, leading to different concrete computational assumptions.

3.1 A Statistically Hiding Noisy Encoding

- **Encoding of x,** $\mathsf{Enc}_n^{\mathcal{R}}(\mathsf{id}, x)$: Here $x \in \mathcal{R}_{\mathsf{id}}$; n is a parameter of the encoding.

[10] This is needed only in the constructions which rely on concrete computational assumptions. A computationally-unbounded ring oracle can be used by the adversary to break the underlying assumption.

- Denote $\mathcal{R}_{\mathsf{id}}$ by R.
- Pick a "pattern" $\sigma \in \{0,1\}^n$.
- Pick a random vector $u \in R^n$ conditioned on $\sum_{i=1}^n u_i = x$.
- Pick a pair of random vectors $(v^0, v^1) \in R^n \times R^n$, conditioned on $v_i^{\sigma_i} = u_i$. That is, the vector u is "hidden" in the pair of vectors v^0 and v^1 according to the pattern σ.
- Output (v^0, v^1, σ).

The encoding could be seen as consisting of two parts (v^0, v^1) and σ, where the latter is information that will allow one to decode this code. For $x \in R$, let $\mathcal{S}_x^{R,n}$ denote the distribution of the first part of the encoding $\mathsf{Enc}_n^{\mathcal{R}}(\mathsf{id}, x)$, namely (v^0, v^1).

This simple encoding has the useful property that it statistically hides x when the decoding information σ is removed. The proof of this fact makes use of the Leftover Hash Lemma [ILL89] (similarly to previous uses of this lemma in [IN96, IKOS06]).

Lemma 1 *Let $n > \log|R| + k$. Then, for all $x \in R$, the statistical distance between the distribution of $\mathcal{S}_x^{R,n}$ and the uniform distribution over $R^n \times R^n$ is $2^{-\Omega(k)}$.*

PROOF: Consider the hash function family $\mathcal{H}$ that consists of functions $H_{v^0,v^1} : \{0,1\}^n \to R$, where $(v^0, v^1) \in R^n \times R^n$, defined as $H_{v^0,v^1}(\sigma) := \sum_i v_i^{\sigma_i}$. It is easily verified that this is a 2-universal hash function family. Then, by the Leftover Hash Lemma,

$$\frac{1}{|\mathcal{H}|}\sum_{H\in\mathcal{H}} \Delta(H(\mathcal{U}_{\{0,1\}^n}), \mathcal{U}_R) = 2^{-\Omega(n-\log|R|)},$$

where $\mathcal{U}_{\{0,1\}^n}$ stands for the uniform distribution over $\{0,1\}^n$ and Δ denotes the statistical difference between two distributions.

To prove the lemma we make use also of the following symmetry between all the possible outcomes of the hash functions: There is a family of permutations on $\mathcal{H}$, $\{\pi_\alpha | \alpha \in R\}$ such that for all $z \in R$, $\Pr[z|H] = \Pr[z+\alpha|\pi_\alpha(H)]$ (where $\Pr[z|H]$ is a shorthand for $\Pr_{\sigma\leftarrow\{0,1\}^n}[H(\sigma) = z]$). In particular we can set $\pi_\alpha(H_{v^0,v^1}) := H_{u^0,u^1}$ where u^0 (respectively u^1) is identical to v^0 (respectively v^1) except for the first co-ordinate which differs by α: $u_1^0 - v_1^0 = u_1^1 - v_1^1 = \alpha$. Then,

$$\begin{aligned}
\frac{1}{|\mathcal{H}|}\sum_{H\in\mathcal{H}} \Delta(H(\mathcal{U}_{\{0,1\}^n}), \mathcal{U}_R) &= \frac{1}{|\mathcal{H}|}\frac{1}{2}\sum_{H\in\mathcal{H}}\sum_{z\in R}\left(|\Pr[z|H] - \frac{1}{|R|}|\right) \\
&= \frac{1}{|\mathcal{H}|}\frac{1}{2}\sum_{z\in R}\sum_{H\in\mathcal{H}}\left(|\Pr[x|\pi_{x-z}(H)] - \frac{1}{|R|}|\right) \\
&= \frac{|R|}{|\mathcal{H}|}\frac{1}{2}\sum_{H\in\mathcal{H}}\left(|\Pr[x|H] - \frac{1}{|R|}|\right) \quad \text{because } \pi_{x-z} \text{ is a permutation} \\
&= \frac{1}{2}\sum_{H\in\mathcal{H}}\left(|\Pr[H|x] - \frac{1}{|\mathcal{H}|}|\right) \quad \text{because with } \Pr[H] = \frac{1}{|\mathcal{H}|}, \Pr[x] = \frac{1}{|R|}.
\end{aligned}$$

Note that the last expression is indeed the statistical difference between $\mathcal{S}_x^{R,n}$ and $\mathcal{U}_{R^n\times R^n}$. To complete the proof note that we have already bounded the first quantity by $2^{-\Omega(n-\log|R|)}$. □

3.2 Linear Code Based Encodings

We describe an abstract noisy encoding scheme for a ring family $\mathcal{R}$. The encoding scheme is specified using *a code generation algorithm* $\mathcal{G}$:

- $\mathcal{G}$ is a randomized algorithm such that $\mathcal{G}^{\mathcal{R}}(\mathsf{id})$ outputs (G, H, L) where G is an $n \times k$ matrix, $L \subseteq [n]$, $|L| = \ell$ and H is another matrix. We note that only G and L will be used in the noisy encoding process; H will be useful in describing the decoding process.

Here k is the security parameter as well as the code dimension, and $n(k)$ (code length) and $\ell(k)$ (number of coordinates *without* noise) are parameters of $\mathcal{G}$. In our instantiations n will be a constant multiple of k and in most cases we will have $\ell = k$.

Let $\mathcal{R}$ be a ring family and $R = \mathcal{R}_{\mathsf{id}}$ from some id. Given $\mathcal{G}$, a parameter $t(k) \leq k$ (number of ring elements to be encoded, $t = 1$ by default), and $x \in R^t$, we define a distribution $\mathcal{E}^{R}_{(\mathcal{G},t)}(x)$, as that of the public output in the following encoding process:

- *Encoding* $\mathsf{Encode}^{\mathcal{R}}_{(\mathcal{G},t)}(\mathsf{id}, x)$*:*
  - Input: $x = (x_1, \ldots, x_t) \in R^t$.
  - Let $(G, L, H) \leftarrow \mathcal{G}^{\mathcal{R}}(\mathsf{id})$
  - Pick a random vector $u \in R^k$ conditioned on $u_i = x_i$ for $i = 1, \ldots, t$ (i.e., u is x padded with $k - t$ random elements). Compute $Gu \in R^n$.
  - Pick a random vector $v \leftarrow R^n$, conditioned on $v_i := (Gu)_i$ for $i \in L$.
  - Let the private output be (G, L, H, v) and the public output be (G, v) (where each ring element is represented as a bit string obtained by the mapping label used by $\mathcal{R}$).

The matrix H is not used in the encoding above, but will be required for a decoding procedure that our protocols will involve. In our main instantiations H can be readily derived from G and L. But we include H explicitly in the outcome of $\mathcal{G}$, because in some cases it is possible to obtain efficiency gains if (G, H, L) are sampled together. We sketch one such case when we describe "Ring code based encoding" in Section 3.2.1.

Assumption 1 (Generic version, for a given $\mathcal{G}$, $\mathcal{R}$ and $t(k)$.) *For all sequences $\{(\mathsf{id}_k, x_k, y_k)\}_k$, let $R_k = \mathcal{R}_{\mathsf{id}_k}$, and suppose $x_k, y_k \in R_k^{t(k)}$. Then the ensembles $\{\mathcal{E}^{R_k}_{(\mathcal{G},t)}(x)\}_k$ and $\{\mathcal{E}^{R_k}_{(\mathcal{G},t)}(y)\}_k$ are computationally indistinguishable.*

For the sake of reference to some previously studied assumptions, we also define a simpler (but stronger) generic assumption, which implies the above version:

Assumption 2 (Generic pseudorandomness version, for a given $\mathcal{G}$ and $\mathcal{R}$.) *For any sequence $\{\mathsf{id}_k\}_k$, let $R_k = \mathcal{R}_{\mathsf{id}_k}$. Then the ensembles $\{\mathcal{E}^{R_k}_{(\mathcal{G},t)}(0^{t(k)})\}_k$ and $\{\big(G \leftarrow \mathcal{G}^{R_k}, v \leftarrow R_k^n\big)\}_k$ are computationally indistinguishable.*

3.2.1 Instantiations of the Encoding

The above generic encoding scheme can be instantiated by specifying a code generation algorithm $\mathcal{G}$, a ring family, and the parameter $t(k)$ which specifies the length of the input to be encoded. We consider three such instantiations.

Random code based instantiation. Our first instantiation of the generic encoding has $t(k) = 1$ and uses a code generation algorithm $\mathcal{G}_{\mathsf{Rand}}$ based on a random linear code. Here the ring family is any field family $\mathcal{F}$. $\mathcal{G}^{\mathcal{F}}_{\mathsf{Rand}}(\mathsf{id})$ works as follows:

- Let $k = |\mathsf{id}|$. Let $n = 2k$ and $\ell = k$. Denote $\mathcal{F}_{\mathsf{id}}$ by F.
- Pick a random $n \times k$ matrix $G \leftarrow F^{n \times k}$.
- Pick a random subset $L \subseteq [n]$, $|L| = k$, such that the $k \times k$ submatrix $G|_L$ is non-singular, where $G|_L$ consists of those rows in G whose indices are in L.[11]
- Let H be the $k \times k$ matrix such that $HG|_L = I$, the $k \times k$ identity matrix. (This H will be used in our protocol constructions.)

The following variants of this instantiation are also interesting:

- Instead of choosing $n(k) = 2k$, we can choose $n(k) > 2k + \log^c |F|$ for some $c < 1$ (say $c = \frac{1}{2}$). By choosing a larger n we essentially weaken the required assumption. (We remark that the case of $n(k) > \log |F|$ is not of much interest to us here, because then our construction which employs this assumption is bettered by our unconditional construction.)
- The above encoding can be directly used with a pseudo-field family instead of a field family. Note that the invertibility of elements was used in deriving H, but in a pseudo-field, except with negligible probability this derivation will still be possible.

Ring code based instantiation. Our next instantiation also has $t(k) = 1$. It uses a code generation algorithm $\mathcal{G}_{\mathsf{Ring}}$ that works with any arbitrary ring family (not just fields). But for simplicity we will assume that the ring has a multiplicative identity 1.[12] Here again in the noisy encoding we will use $t = 1$. $\mathcal{G}^{\mathcal{R}}_{\mathsf{Ring}}(\mathsf{id})$ works follows.

- Let $k = |\mathsf{id}|$. Let $n = 2k$ and $\ell = k$. Denote $\mathcal{R}_{\mathsf{id}}$ by R.
- Pick two $k \times k$ random matrices A and B with elements from R, conditioned on them being upper triangular and having 1 in the main diagonal. Let G be the $2k \times k$ matrix $\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]$.
- Define L as follows. Let $L = \{a_1, \ldots, a_k\}$ where $a_i = i$ or $k + i$ uniformly at random. (That is a_i indices the i-th row in either A or B.)
- Note that $G|_L$ is an upper triangular matrix with 1 in the main diagonal. It is easy to compute an upper triangular matrix H (also with 1 in the main diagonal) using only the ring operations on elements in $G|_L$ such that $HG|_L = I$.

Here, instead of choosing two matrices, we could choose several, to make the resulting assumption weaker at the expense of increasing n.

We point out an alternate encoding which would also work with arbitrary rings. One can construct $G|_L$ and H such that $HG|_L = I$ simultaneously by taking a two *opposite* random walks in the special linear

[11] For efficiency of $\mathcal{G}$, it is enough to try random subsets $L \subseteq [n]$ and check if $G|_L$ is non-singular; in the unlikely event that no L is found in k trials, $\mathcal{G}$ can replace G with an arbitrary matrix with a $k \times k$ identity matrix in the first k rows.

[12] Rings which do not have 1 can be embedded into a ring of double the size which does have 1, by including new elements $a + 1$ for every element a in the original ring, and setting $1 + 1 = 0$.

group $\mathrm{SL}(n, R)$ (i.e., the group of $n \times n$ matrices over the ring R, with determinant 1), where each step in the walk consists of adding or subtracting a row from another row, or a column from another column; in the "opposite" walk, the step corresponding to an addition has a subtraction, and the step corresponding to subtraction has an addition. The random walks start from the identity matrix, and will be long enough for the generated matrices to have sufficient entropy. Note that in such a scheme, we need to rely on the code generation algorithm to simultaneously sample (G, L, H), rather than output just (G, L), because matrix inversion is not necessarily easy for all rings.

Reed-Solomon code based instantiation. In our third instantiation of the generic encoding, we will have $t(k)$ to be a constant fraction of k. The code generation algorithm $\mathcal{G}_{\mathsf{RS}}$ is based on the Reed-Solomon code, and will work with any sufficiently large field family $\mathcal{F}$. $\mathcal{G}_{\mathsf{RS}}^{\mathcal{F}}(\mathsf{id})$ works as follows:

- Let $k = |\mathsf{id}|$. Let $n = ck$, for a sufficiently large constant[13] $c > 4$, and $\ell = 2k - 1$. Denote $\mathcal{F}_{\mathsf{id}}$ by F.
- Pick distinct points $\zeta_i \in F$ for $i = 1, \ldots, k$, and $\vartheta_i \in F$, for $i = 1, \ldots, n$ uniformly at random.
- Define the $n \times k$ matrix G so that it extrapolates a degree $k - 1$ polynomial, given by its value at the k points ζ_i, to the n evaluation points ϑ_i. That is, G is such that for any $u \in F^k$, $(Gu)_i = P(\vartheta_i)$ for $i = 1, \ldots, n$, where P is the unique degree $k - 1$ polynomial such that $P(\zeta_i) = u_i$ for $i = 1, \ldots, k$.
- Pick $L \subseteq [n]$ with $|L| = \ell = 2k - 1$ at random.
- Let H be the $k \times 2k - 1$ matrix such that $(Hv_L)_i = Q(\zeta_i)$, where Q is the unique degree $2(k - 1)$ polynomial such that $Q(\vartheta_j) = v_j$ for all $j \in L$.

3.2.2 Instantiations of Assumption 1

Each of the above instantiations of the encoding leads to a corresponding instantiation of Assumption 1. For the sake of clarity we collect these assumptions below.

Assumption 3 *(a)* **[For $\mathcal{G}_{\mathsf{Rand}}$, with $t(k) = 1$.]** *For any computationally efficient field family $\mathcal{F}$, for all sequences $\{(\mathsf{id}_k, x_k, y_k)\}_k$, let $F_k = \mathcal{F}_{\mathsf{id}_k}$, and suppose $x_k, y_k \in F_k$. Then the ensembles $\{\mathcal{E}^{F_k}_{(\mathcal{G}_{\mathsf{Rand}},1)}(x)\}_k$ and $\{\mathcal{E}^{F_k}_{(\mathcal{G}_{\mathsf{Rand}},1)}(y)\}_k$ are computationally indistinguishable.*

(b) **[For $\mathcal{G}_{\mathsf{Ring}}$, with $t(k) = 1$.]** *For any computationally efficient ring family $\mathcal{R}$, for all sequences $\{(\mathsf{id}_k, x_k, y_k)\}_k$, let $R_k = \mathcal{R}_{\mathsf{id}_k}$, and suppose $x_k, y_k \in R_k$. Then the ensembles $\{\mathcal{E}^{R_k}_{(\mathcal{G}_{\mathsf{Ring}},1)}(x)\}_k$ and $\{\mathcal{E}^{R_k}_{(\mathcal{G}_{\mathsf{Ring}},1)}(y)\}_k$ are computationally indistinguishable.*

(c) **[For $\mathcal{G}_{\mathsf{RS}}$, with $t(k) = k/2$.]**[14] *For any computationally efficient field family $\mathcal{F}$, for all sequences $\{(\mathsf{id}_k, x_k, y_k)\}_k$, let $F_k = \mathcal{F}_{\mathsf{id}_k}$, and suppose $x_k, y_k \in F_k^{t(k)}$. Then the ensembles $\{\mathcal{E}^{F_k}_{(\mathcal{G}_{\mathsf{Rand}},t)}(x)\}_k$ and $\{\mathcal{E}^{F_k}_{(\mathcal{G}_{\mathsf{Rand}},t)}(y)\}_k$ are computationally indistinguishable, for $t \leq k/2$.*

[13] We require $c > 4$ so that Assumption 3(c) will not be broken by known list-decoding algorithms for Reed-Solomon codes. $c = 8$ may be a safe choice, with larger values of c being more conservative.

[14] We can make the assumption weaker by choosing smaller values of t, or larger values of n in $\mathcal{G}_{\mathsf{RS}}$.

4 Product-Sharing Secure Against Passive Corruption

In this section we consider the basic two-party functionality $\mathcal{F}_{\text{pdt-shr}}$ described below

- A sends $a \in R$ and B sends $b \in R$ to $\mathcal{F}_{\text{pdt-shr}}$.
- $\mathcal{F}_{\text{pdt-shr}}$ samples two random elements $z^A, z^B \in R$ such that $z^A + z^B = ab$, and gives z^A to A and z^B to B.

When we want to explicitly refer to the ring in which the computation takes place we will write the functionality as $\mathcal{F}^R_{\text{pdt-shr}}$.

We present three protocols based on noisy encodings, with increasing efficiency, but using stronger assumptions, in the OT-hybrid model for this functionality (some of which are restricted to when R is a field). We then present two protocols based on homomorphic encryption. These protocols are secure only against *static* passive corruption. In Appendix A we present a general transformation, that applies to a class of protocols covering all our above protocols, to obtain protocols that are secure against adaptive passive corruption, with erasures.

4.1 A Basic Protocol with Statistical Security

- **Protocol ρ^{OT}.** A holds $a \in R$ and B holds $b \in R$.
  - B randomly encodes b as specified in Section 3.1. i.e., let $(v^0, v^1, \sigma) \leftarrow \mathsf{Enc}^{\mathcal{R}}_n(\mathsf{id}, b)$. Then $\sum_i v^{\sigma_i} = b$.
  - B sends (v^0_i, v^1_i) (for $i = 1, \ldots, n$) to A.
  - A picks a random vector $t \in R^n$ and sets $z^A = \sum_{i=1}^n t_i$; she computes $w^0_i = av^0_i - t_i$ and $w^1_i = av^1_i - t_i$.
  - A and B engage in n instances of $\binom{2}{1}\,\mathsf{OT}$, where in the i^{th} instance A's inputs are (w^0_i, w^1_i) and B's input is σ_i. B receives $w^{\sigma_i}_i$.
  - A outputs z^A. B outputs the sum of all the n elements he received above: i.e., B outputs

$$z^B := \sum_i w^{\sigma_i}_i = \sum_i (av^{\sigma_i} - t_i) = ab - z^A.$$

We will pick $n > \log(|R|) + k$. Then we have the following result.

Lemma 2 *Suppose $n > \log(|R|) + k$. Then protocol ρ^{OT} securely realizes $\mathcal{F}_{\text{pdt-shr}}$ against static passive corruption. The security is statistical.*

PROOF SKETCH: When B is corrupted, it is easy to construct a simulator to obtain perfect security. The more interesting case is when A is corrupted. Then the simulator Sim behaves as follows.

- Send A's input to $\mathcal{F}_{\text{pdt-shr}}$ and obtain z^A in response.
- Set $t \in R^n$ in A's random tape such that $\sum_i t_i = z^A$. Note that A's output will then be z^A.
- Sample an element $\alpha \leftarrow R$ and run the honest program for B using this input. The only message produced by the simulation is a pair of vectors (v^0, v^1).

By Lemma 1, the message produced by the simulator is statistically close to the message produced by B in the real execution (both being statistically close to the uniform distribution over $R^n \times R^n$), and the simulation is statistically indistinguishable from a real execution. □

4.2 Basic Protocol Using Linear Codes for Rings

We improve on the efficiency of the protocol in Section 4.1 by depending on computational assumptions regarding linear codes. One advantage of the protocol in this section is that it does not explicitly depend on the size of the underlying ring. Restricted to fields, this construction can use the code generation $\mathcal{G}_{\mathsf{Rand}}$; for arbitrary rings with unity, the construction can use $\mathcal{G}_{\mathsf{Ring}}$. Note that both coding schemes generate (G, L, H) such that $HG|_L = I$, which is what the protocol depends on. It uses these codes in a noisy encoding with $t = 1$.

- **Protocol σ^{OT}.** A holds $a \in R$ and B holds $b \in R$.
  - B randomly encodes b using $\mathsf{Encode}^R_{(\mathcal{G},1)}(b)$ to get (G, H, L, v) as the private output. (Note that $t = 1$ in the encoding, and $HG|_L = I$.)
  - B sends (G, v) to A.
  - A picks a random vector $x \in R^k$ and sets $w = av - Gx$.
  - A and B engage in an $\binom{n}{k}$-OT where A's inputs are $(w_1, \ldots, w_n)$ and B's input is L. B receives w_i for $i \in L$. (Recall that when considering passive corruption, an $\binom{n}{k}$-OT maybe implemented using n instances of $\binom{2}{1}$-OT. Here OT is a string-OT and the inputs are labels for the ring elements.)
  - A outputs $z^A := x_1$, the first co-ordinate of x. B outputs $z^B := (Hw_L)_1 = ab - x_1$.

Lemma 3 *If Assumption 1 holds for a code generation scheme $\mathcal{G}$, with $t = 1$, then Protocol σ^{OT} securely realizes $\mathcal{F}_{\mathsf{pdt\text{-}shr}}$, against static passive corruption.*

PROOF SKETCH: The interesting case is when A is corrupt and B is honest. Then the simulator Sim behaves as follows.

- Send A's input to $\mathcal{F}_{\mathsf{pdt\text{-}shr}}$ and obtain z^A in response.
- Set $x \in R^n$ in A's random tape conditioned on $x_1 = z^A$. Note that A's output will then be z^A.
- Sample an element $\alpha \in R$ and run the honest program for B using this input. The only message produced by the simulation is the pair (G, v).

(G, v) is the only message output by (simulated) B in the (simulated) protocol. In the real execution this message is distributed according to $\mathcal{E}^R_{(\mathcal{G},1)}(b)$ whereas in the simulation it is distributed according to $\mathcal{E}^R_{(\mathcal{G},1)}(\alpha)$. By the assumption in the lemma, we conclude that these two distributions are indistinguishable (even if b and α are known), and hence the view of the environment in the real execution is indistinguishable from that in the simulated execution. □

4.3 Amortization using Packed Encoding

In this section we provide a passive-secure protocol in the OT-hybrid model for *multiple instances of* the basic two-party functionality $\mathcal{F}_{\text{pdt-shr}}$. That is, we realize the two-party functionality $\mathcal{F}_{\text{pdt-shr}}^{F^t}$ which takes as inputs $\mathbf{a} \in F^t$ and $\mathbf{b} \in F^t$, and outputs random vectors z^A and z^B to A and B respectively, such that $z^A + z^B = \mathbf{ab} := (a_1 b_1, \ldots, a_t b_t)$ (note that multiplication in F^t refers to coordinate-wise multiplication).

We use the noisy encoding scheme with the code generation algorithm $\mathcal{G}_{\mathsf{RS}}$. We shall choose $t = k/2$.

- **Protocol τ^{OT}.** A holds $\mathbf{a} = (a_1, \ldots, a_t) \in F^t$ and B holds $\mathbf{b} = (b_1, \ldots, b_t) \in F^t$.
  - B randomly encodes x using $\mathsf{Encode}^R_{(\mathcal{G}_{\mathsf{RS}},t)}(x)$ to get (G, H, L, v) as the private output.
  - B sends G and $v = (v_1, \ldots, v_n) \in F^n$ to A. Recall that for some degree $k-1$ polynomial P_b, $v_i := P_b(\vartheta_i)$ for $i \in L$ (and v_i is a random field element if $i \notin L$).
  - Note that the points ϑ_i and ζ_i are implicitly specified by G. A picks a random degree $k-1$ polynomial P_a such that $P_a(\zeta_i) = a_i$ for $i = 1, \ldots, t$, and also a random degree $2(k-1)$ polynomial P_r. A computes $w_i := P_a(\vartheta_i) v_i - P_r(\vartheta_i)$ for $i = 1, \ldots, n$.
  - A and B engage in a $\binom{n}{2k-1}$ OT, where A's inputs are $(w_1, \ldots, w_n)$ and B's input is L. B receives w_i for $i \in L$.
  - B computes $Hw|_L$. Note that then $(Hw|_L)_i = Q(\zeta_i)$ where Q is the unique degree $2(k-1)$ polynomial Q such that $Q(\vartheta_i) = w_i$ for $i \in L$.
  - A sets $z_i^A := P_r(\zeta_i)$ for $i = 1, \ldots, t$. and B sets $z_i^B := Q(\zeta_i)$ for $i = 1, \ldots, t$.
    Note that (if A and B are honest), Q is the degree $2(k-1)$ polynomial $P_a P_b - P_r$, and hence $z_i^A + z_i^B = P_a(\zeta_i) P_b(\zeta_i) = a_i b_i$.
  - A outputs $z^A := (z_1^A, \ldots, z_t^A)$ and B outputs $z^B := (z_1^B, \ldots, z_t^B)$.

Remark about computational efficiency. The computational complexity of Protocol τ^{OT} (ignoring the use of OT) is dominated by the evaluation and interpolation of polynomials (note that the matrices G and H can be stored in an implicit form just by storing the points ϑ_i and ζ_i). As such, in general the complexity would be $O(k \log^2 k)$ for randomly chosen evaluation points [vzGG99]. We note, however, that this complexity can be reduced to $O(k \log k)$ by a more careful selection of evaluation points [vzGG99], at the expense of having to assume that Assumption 3(c) holds also with respect to this specific choice of evaluation points.

Lemma 4 *If Assumption 3(c) holds, then Protocol τ^{OT} securely realizes $\mathcal{F}_{\text{pdt-shr}}^{F^t}$, against static passive corruption.*

PROOF SKETCH: The interesting case is when A is corrupt and B is honest. Then the simulator Sim behaves as follows.

- Send A's input $\mathbf{a}$ to $\mathcal{F}_{\text{pdt-shr}}^{F^t}$ and obtain z^A in response.
- Set A's random tape so that she picks P_r such that $P_r(\zeta_i) = z_i^A$ for $i = 1, \ldots, t$. Note that A's output will then be z^A.
- Sample $\boldsymbol{\alpha} \in F^t$ and run the honest program for B using this input. The only message produced by the simulation is the vector v.

Indistinguishability of the simulation follows because of the assumption in the lemma: given $\boldsymbol{\alpha}$ and $\mathbf{b}$, $\mathcal{E}^R_{(\mathcal{G}_{\mathsf{RS}},t)}(\boldsymbol{\alpha})$ and $\mathcal{E}^R_{(\mathcal{G}_{\mathsf{RS}},t)}(\mathbf{b})$ are computationally indistinguishable. □

4.4 Protocols based on Homomorphic Encryption

In this section, we construct protocols (secure against passive adversaries) for the basic two-party functionality $\mathcal{F}_{\mathsf{pdt\text{-}shr}}$, based on homomorphic encryption. Since we work in the context of rings, by homomorphic encryption (informally speaking), we mean an encryption scheme where it is possible to both: (1) given encryptions of two ring elements x and y, it is possible to generate an encryption of $x + y$; and (2) given a ring element α and an encryption of a ring element x, it is possible to generate an encryption of αx. It is important to stress two points:

- Any encryption scheme that is *group-homomorphic* for the standard representation of the (additive) group $\mathbb{Z}_m$ is immediately homomorphic in our sense with respect to the *ring* $\mathbb{Z}_m$.
- As such, our notion of homomorphic encryption, even though it is defined in the context of rings, *is different from and should not be confused with* the notion of "fully" or "doubly" homomorphic encryption. In particular, we do not require that given encryptions of two ring elements x and y, it is possible to generate an encryption of $x \cdot y$, where $\cdot$ is the ring multiplication operation.

Note that while most homomorphic encryption schemes from the literature fit this definition (since they are group-homomorphic for the standard representation of the (additive) group $\mathbb{Z}_m$), some do not; for example, the El Gamal encryption scheme is group-homomorphic for a subgroup of $\mathbb{Z}_p^*$, but there does not seem to be any ring structure for which El Gamal encryption would be homomorphic in our sense[15].

Furthermore, we consider two types of homomorphic encryption schemes. Informally speaking, the issue that separates these two types of homomorphic encryption schemes is whether the ring underlying the homomorphic encryption scheme can be specified beforehand (which we call a "controlled ring" scheme), or whether it is determined by the key generation algorithm (which we call an "uncontrolled ring"). For example, the key generation algorithm of the classic Goldwasser-Micali encryption scheme [GM84] based on quadratic residuosity always produces keys for a $\mathbb{Z}_2$-homomorphic encryption scheme, and is thus a "controlled ring" scheme. Note that by considering higher residuosity classes, Benaloh [Ben87] similarly constructs "controlled ring" homomorphic encryption schemes for the rings $\mathbb{Z}_p$, where p is a polynomially bounded (small) prime number. On the other hand, schemes like the Paillier cryptosystem [Pai99] are homomorphic with respect to the ring $\mathbb{Z}_n$, where n is a randomly chosen product of two large primes chosen at the time of key generation; n cannot be specified ahead of time. Thus, the Paillier scheme is an example of an "uncontrolled ring" homomorphic encryption scheme.

We first describe formally what we call "controlled ring" homomorphic encryption:

Definition 4.1 *A* "controlled ring" homomorphic encryption *scheme corresponding to a concrete ring family $\mathcal{R}$ is a tuple of algorithms (G, E, D, C), such that:*

1. *(G, E, D) is a semantically secure public-key encryption scheme, except that the algorithm G takes as input both 1^k and id, and the set of values that can be encrypted using the public-key output by G are the elements of $\mathcal{R}_{\mathsf{id}}$.*

[15] Since $\mathbb{Z}_p^*$ is cyclic, it can be associated with the ring $\mathbb{Z}_{p-1}$; however there does not seem to be any computationally efficient way to consider El Gamal encryption to be homomorphic for any nontrivial subring of this ring, as it would seem to require computing discrete logarithms in $\mathbb{Z}_p^*$ or its subgroups.

2. *For any $x_1, x_2 \in \mathcal{R}_{\mathsf{id}}$, given $(pk, sk) \leftarrow G(1^k, \mathsf{id})$ and two ciphertexts $c_1 = E(pk, x_1)$ and $c_2 = E(pk, x_2)$, we have that $C(pk, c_1, c_2)$ outputs a distribution whose statistical distance to the distribution $E(pk, x_1 + x_2)$ is negligible in k.*

3. *For any $x, \alpha \in \mathcal{R}_{\mathsf{id}}$, given $(pk, sk) \leftarrow G(1^k, \mathsf{id})$ and a ciphertext $c = E(pk, x)$, we have that $C(pk, c, \alpha)$ outputs a distribution whose statistical distance to the distribution $E(pk, x \cdot \alpha)$ is negligible in k.*

Such controlled ring homomorphic encryption schemes immediately give rise to a protocol for our basic two-party functionality $\mathcal{F}_{\mathsf{pdt\text{-}shr}}$, as we now demonstrate.

- **Protocol θ.** A holds $a \in \mathcal{R}_{\mathsf{id}}$ and B holds $b \in \mathcal{R}_{\mathsf{id}}$.
  - (Initialization) A runs $G(1^k, \mathsf{id})$ to obtain (pk, sk). This is done only once, as the same public key can be used as many times as necessary.
  - A computes $c = E(pk, a)$, and sends c to B.
  - B chooses $r \in \mathcal{R}_{\mathsf{id}}$ at random, computes $c' = E(pk, r)$, and then computes $c'' = C(pk, C(pk, c, b), c')$ and sends c'' to A. Note that c'' is an encryption of $ab + r$. B outputs $-r$.
  - A computes $v = D(sk, c'')$, and outputs v.

The correctness and privacy properties of this protocol (against passive corruptions) follow immediately from the definition of controlled ring homomorphic encryption.

As mentioned above, unfortunately many known homomorphic encryption schemes do not allow complete control over the ring underlying the homomorphic encryption scheme, and so they do not satisfy the definition of controlled ring homomorphic encryption schemes. We deal with these types of homomorphic encryption schemes separately below.

Definition 4.2 *An* "uncontrolled ring" homomorphic encryption *scheme corresponding to a concrete ring family $\mathcal{R}$ is a tuple of algorithms (G, E, D, C), such that:*

1. *(G, E, D) is a semantically secure public-key encryption scheme, except that the algorithm G outputs id along with the public and private keys, and the set of values that can be encrypted using the public-key output by G are the elements of $\mathcal{R}_{\mathsf{id}}$. Furthermore, it is guaranteed that $|\mathcal{R}_{\mathsf{id}}| > 2^k$, and $|\mathcal{R}_{\mathsf{id}}| < 2^{qk}$ for some universal constant q.*

2. *Given $(pk, sk, \mathsf{id}) \leftarrow G(1^k)$, for any $x_1, x_2 \in \mathcal{R}_{\mathsf{id}}$, and given two ciphertexts $c_1 = E(pk, x_1)$ and $c_2 = E(pk, x_2)$, we have that $C(pk, c_1, c_2)$ outputs a distribution whose statistical distance to the distribution $E(pk, x_1 + x_2)$ is negligible in k.*

3. *Given $(pk, sk, \mathsf{id}) \leftarrow G(1^k)$, for any $x, \alpha \in \mathcal{R}_{\mathsf{id}}$, given a ciphertext $c = E(pk, x)$, we have that $C(pk, c, \alpha)$ outputs a distribution whose statistical distance to the distribution $E(pk, \alpha \cdot x)$ is negligible in k.*

In the case of uncontrolled ring homomorphic encryption schemes, we will not consider general rings, but rather focus our attention on the special case of $\mathbb{Z}_M$ (i.e. $\mathbb{Z}/M\mathbb{Z}$). Here, we will assume that we are using the standard representation of this ring (as integers in $[0, M-1]$ working modulo M). We note that this is our only protocol where a specific representation of the underlying ring is important and required for our result. In this case, using a little bit of standard additional machinery, we can once again construct a quite simple protocol for our basic two-party functionality $\mathcal{F}_{\mathsf{pdt\text{-}shr}}$, as a show below.

- **Protocol ψ.** A holds $a \in \mathbb{Z}_M$ and B holds $b \in \mathbb{Z}_M$.
  - (Initialization) Let $k' = \lceil 2 \log M \rceil + 2 + k$. A runs $G(1^{k'})$ to obtain (pk, sk, N), where $N > 4(2^k M^2)$. This is done only once, as the same public key can be used as many times as necessary.
  - A computes $c = E(pk, a)$, and sends c to B.
  - B chooses $r \in \mathbb{Z}_M$ and $s \in \mathbb{Z}_{2(2^k M)}$ at random, computes $c' = E(pk, r)$, $c'' = E(pk, sM)$, and then computes c''' using the algorithm C repeatedly so that c''' is an encryption of $ab + r + sM$. Note that $ab + r + sM < N$, by choice of parameters. B then sends c''' to A, and outputs $-r \mod M$.
  - A computes $v = D(sk, c''')$, and outputs $v \mod M$.

A straightforward counting argument shows that for any $a, b, r \in \mathbb{Z}_M$, setting $w = ab + r \mod M$, we have that the statistical distance between the distributions $D_1 = (ab + r + sM)$ and $D_2 = (w + sM)$, where $s \in \mathbb{Z}_{2(2^k M)}$ is chosen at random, is at most 2^{-k}. This is because $ab + r \leq 2M^2$, and so there are at most $2M$ choices of s for which $w + sM$ would not be in the support of D_1. Thus, by the definition of uncontrolled ring homomorphic encryption, the correctness and privacy properties of this protocol (against passive corruptions) follow immediately.

Matrix rings. Although we focus on the case of $\mathbb{Z}_M$ above, it is easy to see that this approach can generalized to other related settings, such as the ring of n by n matrices over $\mathbb{Z}_M$, in a straightforward manner. At a high level, this is because any $\mathbb{Z}_M$-homomorphic encryption scheme immediately gives rise to an encryption scheme that is homomorphic for the ring of n by n matrices over $\mathbb{Z}_M$. In this context, by simply encrypting each entry in the matrix, the homomorphic property of matrix addition would follow immediately from the homomorphic property with respect to addition of the underlying encryption scheme. The slightly interesting case is the "scalar" multiplication (by a known matrix) property of the homomorphic encryption scheme. It is easy to see that this property also holds, since each entry of the product matrix is just a degree-2 function of the entries of the two matrices being multiplied. Thus, for instance in our case of n by n matrices, one can compute the $\mathcal{F}_{\text{pdt-shr}}$ functionality with only $O(n^2)$ ciphertexts communicated, even though no algebraic circuits for matrix multiplication are known (or generally believed to exist) with $O(n^2)$ gates.

The discussion regarding matrices above is implicitly written in the context of controlled-ring homomorphic encryption. In the context of uncontrolled-ring homomorphic encryption, using the same ideas, Protocol ψ can be directly adapted to allow one to compute m degree-2 functions over n variables while communicating only $O(m+n)$ ciphertexts. This allows one to use uncontrolled-ring homomorphic encryption to compute the $\mathcal{F}_{\text{pdt-shr}}$ functionality for n by n matrices over $\mathbb{Z}_M$ with only $O(n^2)$ ciphertexts (for an encryption scheme over $\mathbb{Z}_N$ where $\log N$ is $O(\log n + k + \log M)$) being communicated.

5 General Arithmetic Computation against Active Corruption

As already discussed in Section 1.3, our general protocols are obtained by applying the general technique of [IPS08], with appropriate choices of the "outer protocol" and the "inner protocol" that apply to the arithmetic setting.

More concretely, the result from [IPS08] shows how obtain a UC-secure protocol in the OT-hybrid model for any (probabilistic polynomial time) two-party functionality f against active corruption by making a *black-box* use of the following two ingredients:

1. an "outer protocol" for f which employs k auxiliary parties (servers); this protocol should be UC-secure against active corruption provided that only some constant fraction the servers can be corrupted; and

2. an "inner protocol" for implementing a reactive two-party functionality ("inner functionality") corresponding to the local computation of each server, in which the server's state is secret-shared between Alice and Bob. In contrast to the outer protocol, this protocol only needs to be secure against *passive* corruption. The inner protocol can be implemented in the OT-hybrid model.

While the general result of [IPS08] is not sensitive to the type of secret sharing used for defining the inner functionality, in our setting it is crucial that any ring elements stored by a server will be secret-shared between Alice and Bob using additive secret sharing over the ring. Given our protocols for $\mathcal{F}_{\text{pdt-shr}}$, this will let us have the the inner protocol use the ring in a black-box fashion, as described below.

Note that the only operations that the server in an outer protocol needs to do one of the following operations: add two ring elements, multiply two ring elements, sample a ring element uniformly at random, or check if two ring elements are equal. If there are oprations which do not involve any ring elements, the inputs and outputs to these operations are maintained as bit strings and an arbitrary protocol for boolean circuit evaluation (e.g., GMW in the OT-hybrid model) can be employed. Among the operations that do involve ring elements, addition and sampling are straightforward: whenever a server in the outer protocol needs to locally add two ring elements x, y, this can be done locally in the inner protocol by having each of Alice and Bob add their local shares of the two secrets. When a server in the outer protocol needs to sample a random ring element, Alice and Bob locally sample the shares of this element. For multiplication, when a server needs to multiply two ring elements x, y in the outer protocol, the inner protocol will need to apply a sub-protocol for the following two-party functionality:

- A holds x_A and y_A, B holds x_B and y_B.
- The server should compute random values c_A and c_B such that $c_A + c_B = (x_A + x_B)(y_A + y_B)$.
- A is given c_A and B is given c_B.

The above functionality can be realized (in the semi-honest model) by making two calls to any of the product-sharing protocols from Section 4. Specifically, a secure reduction from the above functionality to $\mathcal{F}_{\text{pdt-shr}}$ may proceed as follows:

- A and B engage in two instances of $\mathcal{F}_{\text{pdt-shr}}$ with inputs (x_A, y_B) and (y_A, x_B) and obtain (α_A, α_B) and (β_A, β_B) where $\alpha_A + \alpha_B = x_A y_B$ and $\beta_A + \beta_B = y_A x_B$.
- A outputs $c_A := x_A y_A + \alpha_A + \beta_A$ and B outputs $c_B := x_B y_B + \alpha_B + \beta_B$.

There will be several such instances of $\mathcal{F}_{\text{pdt-shr}}$ in each round. Note that Protocol τ^{OT} can be used to realize multiple instances of $\mathcal{F}_{\text{pdt-shr}}$ with a constant amortized algebraic complexity per instance.

The final type of computation performed by servers involving ring elements is equality check between two ring elements. In all the outer protocols we employ, the result of such an equality test is made public. (In fact, in our setting of "security with abort," the outer protocols we consider will abort whenever an inequality is detected by an honest server in such an equality test.) The corresponding inner functionality needs to check that $x_a + x_b = y_a + y_b$, where x_a, y_a are identifiers of ring elements known to Alice and x_b, y_b are known to Bob. One way to do this would be by letting Alice locally compute $x_a - y_a$, Bob locally compute $y_b - x_b$, and then using an arbitrary inner protocol for boolean circuits for comparing the two

identifiers. This relies on our assumption that each ring element has a unique identifier. However, in fact in the outer protocols we consider, there is a further structure that allows us to avoid this generic approach. The elements to be compared by a server in our outer protocols will always be known to one of the parties (Alice or Bob), and hence in a passive-secure implementation this comparison can be done locally by that party. (This is referred to as a "type I computation" in [IPS08]. Note that given a passive-secure implementation, the compiler of [IPS08] ensures over all security.)

Below we summarize the results we obtain by combining appropriate choices for the outer protocol with the inner protocols obtained via the shared-product protocols from Section 4. All these results can be readily extended to the multi-party setting as well, where the complexity grows polynomially with the number of parties; see Appendix B.

Unconditionally secure protocol. To obtain our unconditional feasibility result for black-box rings, we use the protocol from [CFIK03] (which makes a black-box use of an arbitrary ring) as the outer protocol and the unconditional protocol ρ^{OT} to build the inner protocol. This yields the following result:

Theorem 1 *For any arithmetic circuit C, there exists a protocol Π in the OT-hybrid model that is a secure black-box realization of C-evaluation for the set of all ring families. The security holds against adaptive corruption with erasures, in computationally unbounded environments.*

The *arithmetic* communication complexity of the protocol ρ^{OT}, and hence that of the above protocol, grows linearly with (a bound on) $|\log \mathcal{R}_{\mathsf{id}}|$. (Recall that, by convention, the required upper bound is given by $|\mathsf{id}|$; otherwise such a bound can be inferred from the length of identifiers.)

Protocols from noisy encodings. To obtain a computationally secure protocol whose arithmetic communication complexity is independent of the ring, we shall depend on Assumption 1, instantiated with the code generation algorithm $\mathcal{G}_{\mathsf{Rand}}$ based on random linear codes. By replacing ρ^{OT} by σ^{OT} (with $\mathcal{G}_{\mathsf{Rand}}$ as the code generation scheme) in the previous construction we obtain the following:

Theorem 2 *Suppose that Assumption 3(a) holds. Then, for every arithmetic circuit C, there exists a protocol Π in the OT-hybrid model that is a secure black-box realization of C-evaluation for the set of all computationally efficient field families $\mathcal{F}$. The security holds against adaptive corruption with erasures. Further, the arithmetic complexity of Π is $\mathrm{poly}(k) \cdot |C|$, independently of $\mathcal{F}$ or id.*

Using $\mathcal{G}_{\mathsf{Ring}}$ instead of $\mathcal{G}_{\mathsf{Rand}}$, this result extends to all ring families for which Assumption 1 holds with $\mathcal{G}_{\mathsf{Ring}}$. Recall that we propose this assumption for *all efficient* computational ring families $\mathcal{R}$.

Theorem 3 *Suppose that Assumption 3(b) holds. Then, for every arithmetic circuit C, there exists a protocol Π in the OT-hybrid model that is a secure black-box realization of C-evaluation for the set of all computationally efficient ring families $\mathcal{R}$. The security holds against adaptive corruption, with erasures. Further, the arithmetic complexity of Π is $\mathrm{poly}(k) \cdot |C|$, independently of $\mathcal{R}$ or id.*

Finally, our most efficient protocol will be obtained by using a variant of the protocol from [DI06] as the outer protocol (see Appendix C) and an inner protocol which is based on τ^{OT} (with $n = O(k)$ and $t = \Omega(k)$). To get the specified computational complexity, the size of the field should be super-polynomial in the security parameter. (The communication complexity does not depend on this assumption.)

Theorem 4 *Suppose that Assumption 3(c) holds. Then, for every arithmetic circuit C, there exists a protocol Π in the OT-hybrid model with the following properties. The protocol Π is a secure black-box realization of C-evaluation for the set of all computationally efficient field families $\mathcal{F}$, with respect to all computationally bounded environments for which $|\mathcal{F}_{\mathsf{id}}|$ is super-polynomial in k. The security of Π holds against adaptive*

corruption with erasures. The arithmetic communication complexity of Π *is* $O(|C| + k \cdot \mathsf{depth}(C))$*, where* $\mathsf{depth}(C)$ *denotes the depth of* C*, and its arithmetic computation complexity is* $O(\log^2 k) \cdot (|C| + k \cdot \mathsf{depth}(C))$*. Its round complexity is* $O(\mathsf{depth}(C))$*.*

By using a suitable choice of fields and evaluation points for the Reed-Solomon encoding (see Section 4.3), and under a corresponding specialization of Assumption 3(c), the computational overhead of the above protocol can be reduced from $O(\log^2 k)$ to $O(\log k)$. (In this variant we do not attempt to make a black-box use of the underlying field and rely on the standard representation of field elements.)

Protocols from homomorphic encryption. We also consider protocols which make a black-box[16] use of homomorphic encryption. These are obtained in a manner similar to above, but using protocols θ and ψ as the inner protocols and [CFIK03] as the outer protocol. Using these we obtain the following theorems:

Theorem 5 *For every arithmetic circuit* C*, there exists a protocol* Π *in the* OT*-hybrid model, such that for every ring family* $\mathcal{R}$*, the protocol* $\Pi^{\mathcal{R}}$ *securely realizes* $\mathcal{F}_C^{\mathcal{R}}$ *by making a* black-box *use of any controlled-ring homomorphic encryption for* $\mathcal{R}$*. The security holds against adaptive corruption with erasures. The number of invocations of the encryption scheme is* $\mathrm{poly}(k) \cdot |C|$*, independently of* $\mathcal{R}$ *or* id*.*

Note that the above theorem can be instantiated with the ring of n by n matrices over $\mathbb{Z}_p$, and the communication complexity of the resulting protocol would be $poly(k) \cdot |C| \cdot n^2$. Combined with [MW08], this yields constant-round protocols for secure linear algebra which make a black-box use of homomorphic encryption and whose communication complexity is nearly linear in the input size.

For the case of fields, we obtain the following more efficient version of the result by using the efficient outer protocol from Appendix C:

Theorem 6 *For every arithmetic circuit* C*, there exists a protocol* Π *in the* OT*-hybrid model, such that for every field family* $\mathcal{F}$*, the protocol* $\Pi^{\mathcal{F}}$ *securely realizes* $\mathcal{F}_C^{\mathcal{R}}$ *by making a* black-box *use of any controlled-ring homomorphic encryption for* $\mathcal{F}$*. The security holds against adaptive corruption with erasures. Further,* Π *makes* $O(|C| + k \cdot \mathsf{depth}(C))$ *invocations of the encryption scheme, and the communication complexity is dominated by sending* $O(|C| + k \cdot \mathsf{depth}(C))$ *ciphertexts.*

We also obtain analogous results for uncontrolled-ring homomorphic encryption:

Theorem 7 *For every arithmetic circuit* C *there exists a black-box construction of a protocol* Π *in the* OT*-hybrid model from any uncontrolled-ring homomorphic encryption for the standard representation of the ring family* $\mathbb{Z}_M$*, such that* Π *is a secure realization of* C*-evaluation for the same ring family under the standard representation. The security holds against adaptive corruption with erasures. The number of invocations of the encryption scheme is* $\mathrm{poly}(k) \cdot |C|$*, independently of* id*, and the communication complexity is dominated by* $\mathrm{poly}(k) \cdot |C|$ *ciphertexts. During the protocol, the ring size parameter fed to the encryption scheme by honest parties is limited to* $k' = O(k + |\mathsf{id}|)$*.*

If, further, the ring over which C *should be computed is restricted to be a field, there exists a protocol as above which makes* $O(|C| + k \cdot \mathsf{depth}(C))$ *invocations of the encryption scheme, and where the communication complexity is dominated by sending* $O(|C| + k \cdot \mathsf{depth}(C))$ *ciphertexts.*

The efficient version of the above theorem also applies to the case of arithmetic computation over pseudo-fields, in scenarios where it is computationally hard to find zero divisors. Furthermore, it can be generalized to the ring of n by n matrices, which when used with constructions of uncontrolled-ring $\mathbb{Z}_N$-homomorphic

[16] Here and in the following, when saying that a construction makes a black-box use of a homomorphic encryption primitive we refer to the notion of a fully black-box reduction, as defined in [RTV04]. This roughly means that not only does the construction make a black-box use of the primitive, but also its security is proved via a black-box reduction.

encryption schemes from the literature [Pai99, DJ02] would yield arithmetic protocols for matrices over large rings whose complexity grows quadratically with n.

We finally note that in the *stand-alone* model, the OT oracle in the above protocols can be realized by making a black-box use of the homomorphic encryption primitive without affecting the asymptotic number of calls to the primitive. This relies on the black-box construction from [IKLP06] and the fact that only $O(k)$ OTs need to be secure against active corruption. Thus, the above theorems hold also in the plain, stand-alone model (as opposed to the OT-hybrid UC-model), assuming that the underlying ring has identity.[17]

Acknowledgments. We thank Jens Groth, Farzad Parvaresh, Oded Regev, and Ronny Roth for helpful discussions.

References

[ACS02] Joy Algesheimer, Jan Camenisch, and Victor Shoup. Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 417–432. Springer, 2002.

[AF90] Martín Abadi and Joan Feigenbaum. Secure circuit evaluation. *J. Cryptology*, 2(1):1–12, 1990.

[AIK07] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography with constant input locality. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 92–110. Springer, 2007. Full version in `http://www.cs.princeton.edu/~bappelba`.

[AL07] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 137–156. Springer, 2007.

[BCD+08] Peter Bogetoft, Dan Lund Christensen, Ivan Damgard, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael Schwartzbach, and Tomas Toft. Multiparty computation goes live. Cryptology ePrint Archive, Report 2008/068, 2008. `http://eprint.iacr.org/`.

[Bea95] Donald Beaver. Precomputing oblivious transfer. In Don Coppersmith, editor, *CRYPTO*, volume 963 of *Lecture Notes in Computer Science*, pages 97–109. Springer, 1995.

[Ben87] Josh Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Department of Computer Science, Yale University, 1987.

[BF01] Dan Boneh and Matthew K. Franklin. Efficient generation of shared rsa keys. *J. ACM*, 48(4):702–722, 2001. Earlier version in Crypto '97.

[BFKL93] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291. Springer, 1993.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. 20th STOC*, pages 1–10. ACM, 1988.

[17] The identity element is used in the standard construction of semi-honest OT from homomorphic encryption.

[BKY07] Daniel Bleichenbacher, Aggelos Kiayias, and Moti Yung. Decoding interleaved reed-solomon codes over noisy channels. *Theor. Comput. Sci.*, 379(3):348–360, 2007. Earlier version in ICALP '03.

[BL96] Dan Boneh and Richard J. Lipton. Algorithms for black-box fields and their application to cryptography (extended abstract). In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 283–297. Springer, 1996.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Electronic Colloquium on Computational Complexity (ECCC) TR01-016, 2001. Previous version "A unified framework for analyzing security of protocols" availabe at the ECCC archive TR01-016. Extended abstract in FOCS 2001.

[Can05] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2005. Revised version of [Can01].

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols. In *Proc. 20th STOC*, pages 11–19. ACM, 1988.

[CD01] Ronald Cramer and Ivan Damgård. Secure distributed linear algebra in a constant number of rounds. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 119–136. Springer, 2001.

[CDN01] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In *EUROCRYPT*, pages 280–299, 2001. LNCS No. 2045.

[CF02] Ronald Cramer and Serge Fehr. Optimal black-box secret sharing over arbitrary abelian groups. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 272–287. Springer, 2002.

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *STOC*, pages 639–648, 1996.

[CFIK03] Ronald Cramer, Serge Fehr, Yuval Ishai, and Eyal Kushilevitz. Efficient multi-party computation over rings. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 596–613. Springer, 2003.

[CIK+01] Ran Canetti, Yuval Ishai, Ravi Kumar, Michael K. Reiter, Ronitt Rubinfeld, and Rebecca N. Wright. Selective private function evaluation with applications to private statistics. In *PODC*, pages 293–304, 2001.

[CKP07] Ronald Cramer, Eike Kiltz, and Carles Padró. A note on secure computation of the moore-penrose pseudoinverse and its application to secure linear algebra. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 613–630. Springer, 2007.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party computation. In *Proc. 34th STOC*, pages 494–503. ACM, 2002.

[CS03] Don Coppersmith and Madhu Sudan. Reconstructing curves in three (and higher) dimensional space from noisy data. In *STOC*, pages 136–142. ACM, 2003.

[DF89] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer, 1989.

[DFK+06] Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, and Tomas Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 285–304. Springer, 2006.

[DI06] Ivan Damgård and Yuval Ishai. Scalable secure multiparty computation. In Cynthia Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 501–520. Springer, 2006.

[DJ02] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *CT-RSA*, pages 79–95, 2002.

[DN03] Ivan Damgård and Jesper Buus Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 247–264. Springer, 2003.

[DNO08] Ivan Damgård, Jesper Buus Nielsen, and Claudio Orlandi. Essentially optimal universally composable oblivious transfer. Cryptology ePrint Archive, Report 2008/220, 2008. `http://eprint.iacr.org/`.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[FH96] Matthew K. Franklin and Stuart Haber. Joint encryption and message-efficient secure computation. *J. Cryptology*, 9(4):217–232, 1996.

[FMY98] Yair Frankel, Philip D. MacKenzie, and Moti Yung. Robust efficient distributed rsa-key generation. In *STOC*, pages 663–672, 1998.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2004.

[FY92] Matthew K. Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *STOC*, pages 699–710. ACM, 1992.

[Gil99] Niv Gilboa. Two party rsa key generation. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 116–129. Springer, 1999.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984. Preliminary version appeared in STOC' 82.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play ANY mental game. In ACM, editor, *Proc. 19th STOC*, pages 218–229. ACM, 1987. See [Gol04, Chap. 7] for more details.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.

[Gro08] Jens Groth. Linear algebra with sub-linear zero-knowledge arguments. Manuscript, 2008.

[GS99] Venkatesan Guruswami and Madhu Sudan. Improved decoding of reed-solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

[HIKN08] Danny Harnik, Yuval Ishai, Eyal Kushilevitz, and Jesper Buus Nielsen. OT-combiners via secure computation. In Ran Canetti, editor, *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 393–411. Springer, 2008.

[HL08] Carmit Hazay and Yehuda Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In Ran Canetti, editor, *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 155–175. Springer, 2008.

[IKLP06] Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions for secure computation. In *STOC*, pages 99–108. ACM, 2006.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161. Springer, 2003.

[IKOS06] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography from anonymity. In *FOCS*, pages 239–248. IEEE, 2006.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *STOC*, pages 21–30. ACM, 2007.

[IKOS08] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *STOC*, pages 433–442. ACM, 2008.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstract). In *STOC*, pages 12–24. ACM, 1989.

[IN96] Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *J. Cryptology*, 9(4):199–216, 1996.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, pages 572–591, 2008.

[KMWF07] Eike Kiltz, Payman Mohassel, Enav Weinreb, and Matthew K. Franklin. Secure linear algebra using linearly recurrent sequences. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 291–310. Springer, 2007.

[KY08] Aggelos Kiayias and Moti Yung. Cryptographic hardness based on the decoding of reed-solomon codes. *IEEE Transactions on Information Theory*, 54(6):2752–2769, 2008.

[LO85] J. C. Lagarias and Andrew M. Odlyzko. Solving low-density subset sum problems. *J. ACM*, 32(1):229–246, 1985.

[LP02] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. *J. Cryptology*, 15(3):177–206, 2002. Earlier version in Crypto '00.

[MR07] Ueli M. Maurer and Dominik Raub. Black-box extension fields and the inexistence of field-homomorphic one-way permutations. In *ASIACRYPT*, pages 427–443, 2007.

[MW08] Payman Mohassel and Enav Weinreb. Efficient secure linear algebra in the presence of covert or computationally unbounded adversaries. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 481–496. Springer, 2008.

[NN90] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. In *STOC*, pages 213–223. ACM, 1990.

[NP06] Moni Naor and Benny Pinkas. Oblivious polynomial evaluation. *SIAM J. Comput.*, 35(5):1254–1281, 2006. Earlier version in STOC '99.

[NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *ACM Conference on Electronic Commerce*, pages 129–139, 1999.

[NW06] Kobbi Nissim and Enav Weinreb. Communication efficient secure linear algebra. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 522–541. Springer, 2006.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.

[PS98] Guillaume Poupard and Jacques Stern. Generation of shared rsa keys by two parties. In Kazuo Ohta and Dingyi Pei, editors, *ASIACRYPT*, volume 1514 of *Lecture Notes in Computer Science*, pages 11–24. Springer, 1998.

[PV05] Farzad Parvaresh and Alexander Vardy. Correcting errors beyond the guruswami-sudan radius in polynomial time. In *FOCS*, pages 285–294. IEEE, 2005.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, pages 554–571, 2008.

[Rab81] M. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2004.

[Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11), November 1979.

[Tof07] Tomas Toft. *Primitives and Applications for Multi-party Computation*. PhD thesis, Department of Computer Science, Aarhus University, 2007.

[vzGG99] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999. Earlier version available on `http://www.wisdom.weizmann.ac.il/~oded/frag.html`.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proc. 27th FOCS*, pages 162–167. IEEE, 1986.

A Security Against Adaptive Passive Corruption, with Erasures

Here we present a general transformation, that applies to a class of protocols covering all our protocols from Section 4, to obtain protocols that are secure against *adaptive* passive corruption, in the model with erasures. This transformation is done in a simple way using standard techniques. At a high level, the idea is simply to call our basic protocol on randomly chosen inputs, erase the "local computations" done while executing the basic protocol, and then communicate "corrections" in order to convert the outputs of the random execution into the desired outputs for the real inputs (in a manner very similar to Beaver's reduction of OT to random OT [Bea95]). Intuitively, in our new protocol, if an adversary adaptively corrupts a party during the initial random invocation of the basic protocol, there is no problem since the protocol was anyway run on random inputs chosen independently of the parties' actual inputs (although this is not quite accurate, which is why we introduce a notion of "special simulation" below). On the other hand, if the adversary corrupts a party after the basic protocol is done, then since the party has already erased the local computations of the protocol, we are free to choose a "random-looking" output from the basic protocol in such a way that we can use it to explain the actual inputs and outputs that we have.

The main protocol in this section, $\widetilde{\pi}^{\mathsf{OT}}$ has security against passive adaptive corruption, with erasures. $\widetilde{\pi}^{\mathsf{OT}}$ is built using any protocol π^{OT} with a simpler security property described below. Applying the transformation in this section also has another efficiency advantage in scenarios where pre-processing interaction is possible, and this is discussed briefly in a remark at the end of this section.

A.1 Special Simulation Security Against Passive Corruption

It will be convenient for us to introduce an intermediate notion of security of multi-party computation against *static* passive corruption, which will then enable us to obtain security against *adaptive* passive corruption with erasures. This intermediate security property is quite weak, and is required to hold only against random inputs (though the candidates we shall use later in fact satisfy stronger security).

Let $\mathcal{F}$ be a secure function evaluation functionality. We use the following terminology.

- An environment Env is said to be a *random-input environment* if it provides independent random inputs (according to a specified distribution) to each party.
- A simulator Sim is said to be a *special simulator* if it behaves as follows:
  1. Sim sends the corrupt parties' inputs to $\mathcal{F}$, and obtains the outputs from $\mathcal{F}$.
  2. Sim picks *random inputs for all the honest parties to be simulated*. Sim also sets the random tapes of all the parties (corrupt and honest). These choices are (jointly) indistinguishable from the uniform (or specified) distribution, even given the input of the corrupt parties. (However Sim can correlate these choices with the output obtained from $\mathcal{F}$).
  3. Sim ensures that on interacting with the simulated honest parties, the corrupt parties will produce the *same outputs as given by* $\mathcal{F}$. If this is not the case Sim will abort. Otherwise, it reports the view of the corrupt parties in this execution to the environment.

Definition A.1 *A protocol π is said to* securely realize $\mathcal{F}$ on random inputs, against passive corruption, with special simulation *if there exists a special simulator* Sim *such that for all random-input environments* Env *and a static passive adversary* Adv*, the real execution of the protocol π between the parties is indistinguishable to* Env*, from an ideal execution of the parties interacting with $\mathcal{F}$ and* Sim.

A.2 Special Simulation Security to Security Against Adaptive Corruption with Erasures

Given a protocol π^{OT} which securely realizes $\mathcal{F}_{\mathsf{pdt\text{-}shr}}$ on random inputs, against passive corruption, with special simulation, below we show how to construct a protocol $\widetilde{\pi}^{\mathsf{OT}}$ with security against adaptive passive corruption, with erasures.

- **Protocol $\widetilde{\pi}^{\mathsf{OT}}$.** A holds $a \in R$ and B holds $b \in R$.

  1. A picks $r^A \in R$ and B picks $r^B \in R$ at random.
  2. A and B run π^{OT} with inputs r^A and r^B respectively, and obtains outputs s^A and s^B respectively. Note that $s^A + s^B = r^A r^B$.
  3. A and B erase the memory used for π^{OT}. (They retain the inputs and outputs, namely (r^A, s^A) and (r^B, s^B) respectively.)
  4. A sends $a - r^A$ to B, and B sends $b - r^B$ to A.
  5. A outputs $z^A := a(b - r^B) + s^A$; B outputs $z^B := (a - r^A)r^B + s^B$. Note that $z^A + z^B = ab$.

We now show the following:

Lemma 5 *For any π^{OT} which securely realizes $\mathcal{F}_{\mathsf{pdt\text{-}shr}}$ on random inputs, against passive corruption, with special simulation, protocol $\widetilde{\pi}^{\mathsf{OT}}$ is a secure realization of $\mathcal{F}_{\mathsf{pdt\text{-}shr}}$ against adaptive passive corruption with erasures. If the security of π^{OT} is statistical, so is that of $\widetilde{\pi}^{\mathsf{OT}}$.*

PROOF SKETCH: The interesting cases are when during the protocol initially A is corrupted and later B is corrupted, or when initially B is corrupted and later A is corrupted. (Recall that all corruptions are passive.)

Let $\widetilde{\mathsf{Env}}$ be an arbitrary environment which gives A and B inputs for $\widetilde{\pi}^{\mathsf{OT}}$. We will consider the case when A is corrupted initially and B may be corrupted later. The other case is symmetric for this analysis. Our simulator $\widetilde{\mathsf{Sim}}$ works as follows.

- $\widetilde{\mathsf{Sim}}$ sends a to $\mathcal{F}_{\mathsf{pdt\text{-}shr}}$ and obtains z^A from it.
- $\widetilde{\mathsf{Sim}}$ picks a random value $c \leftarrow R$ and sets $s^A := z^A - ac$, and also picks a random value $r^A \leftarrow R$.
- Next $\widetilde{\mathsf{Sim}}$ internally runs the special simulator Sim for π^{OT} with r^A as input to A. Sim expects to interact with an instance of $\mathcal{F}_{\mathsf{pdt\text{-}shr}}$, which, for clarity, we will denote by $\mathcal{F}'_{\mathsf{pdt\text{-}shr}}$. Sim simulates $\mathcal{F}'_{\mathsf{pdt\text{-}shr}}$ by providing s^A as the output for A.
- If $\widetilde{\mathsf{Env}}$ instructs to corrupt B before this simulation finishes and the erasure step (step 3) is simulated, then $\widetilde{\mathsf{Sim}}$ obtains the simulated state for B in π^{OT} from Sim; then $\widetilde{\mathsf{Sim}}$ constructs a state for B in $\widetilde{\pi}^{\mathsf{OT}}$ by combining this with b, the input to B (which is not used until step 4 of the protocol), and reports this to $\widetilde{\mathsf{Env}}$.
- If B is still not corrupted at step 4, $\widetilde{\mathsf{Sim}}$ uses the value c as the simulated message from B to A in step 4. Note that $z^A = ac + s^A$.
- By this step B has already erased his state during the execution of π^{OT}. So if B is corrupted at any point after this, its state can be explained by giving (b, r^B, s^B): for this the simulator $\widetilde{\mathsf{Sim}}$ will obtain b by corrupting B in the ideal world, and set $r^B := b - c$ and $s^B := r^A r^B - s^A$, where r^A and s^A are the input and output of A in the simulated execution of π^{OT}. Note that this pair (r^B, s^B) is consistent with (r^A, s^A) by the functionality $\mathcal{F}_{\mathsf{pdt\text{-}shr}}$.

The indistinguishability of simulation follows from the two requirements on the simulator Sim for π^{OT}: that it is a special simulator and that it provides an indistinguishable simulation against static corruption (on random inputs).

If B is corrupted before step 4, the simulated execution is indistinguishable from the real execution. To see this, firstly note that Sim is given s^A as the output from $\mathcal{F}'_{\mathsf{pdt\text{-}shr}}$, but this is indeed a random element (because z^A is random). Then, Sim is guaranteed to set the random tape of A, as well as B's input and random tape to be indistinguishable from uniformly random choices. So the simulated state of A and B are indistinguishable from the real execution up to step 4. The simulated state of B is completed by incorporating B's input b (the state used in execution before step 4 being independent of b).

If B is corrupted after step 4, then we consider the following two experiments, with an environment Env which consists of the given environment $\widetilde{\mathsf{Env}}$ as well as part of our simulator which picks c (but does not get z^A or compute s^A). The environment Env provides r^A as input to A and $r^B := b - c$ to B. It outputs the bit output by $\widetilde{\mathsf{Env}}$.

REAL: A and B execute π^{OT} on their inputs from Env.

IDEAL: In the IDEAL execution $\mathcal{F}'_{\mathsf{pdt\text{-}shr}}$ gives a random pair (s^A, s^B) such that $s^A + s^B = r^A r^B$. Sim interacts with Env simulating the internal state of A.

By the security requirement on Sim, the two experiments are indistinguishable to the environment Env. Further the REAL experiment above is identical to the REAL execution of $\widetilde{\pi}^{\mathsf{OT}}$ with $\widetilde{\mathsf{Env}}$. To complete the proof we need to argue that the IDEAL execution above (with Env) is identical to our IDEAL execution (with $\widetilde{\mathsf{Env}}$). Note that in our description of the simulation $s^A := z^A - ac$, whereas in the IDEAL execution with Env, s^A is just a random element. However, though the environment Env knows a and c, z^A will be picked at random (by $\mathcal{F}_{\mathsf{pdt\text{-}shr}}$ in our IDEAL execution). In other words, we could consider a modified $\mathcal{F}'_{\mathsf{pdt\text{-}shr}}$ which receives ac from the environment Env, then picks a random element z^A and sets $s^A := z^A - ac$, without altering the experiment. With this modification, our IDEAL execution (with $\widetilde{\mathsf{Env}}$ and $\widetilde{\mathsf{Sim}}$) is identical to the IDEAL execution with Env and Sim.

□

In Section 4, for the protocols π^{OT}, σ^{OT} and τ^{OT} we showed security against static passive corruption (even for non-random inputs). The simulators we used in these proofs are in fact special simulators. The same is easily seen to be true for protocols θ and ψ based on homomorphic encryption. Thus we have the following result.

Lemma 6 *Protocols π^{OT}, σ^{OT}, τ^{OT}, θ, and ψ securely realize $\mathcal{F}_{\mathsf{pdt\text{-}shr}}$ (or $\mathcal{F}^{F^t}_{\mathsf{pdt\text{-}shr}}$ in the case of Protocol τ^{OT}), on random inputs, against passive corruption, with special simulation.*

Hence, by plugging them into the protocol $\widetilde{\pi}^{\mathsf{OT}}$ we obtain corresponding protocols which are secure against passive, adaptive corruption with erasures.

Remark. The structure of the protocol in this section allows an efficiency gain by employing pre-processing. The first half of the protocol, which is executed on random inputs can be carried out before the actual function evaluation starts. Further, when used to implement a reactive functionality, the entire set of steps involving OT that will be ever used in the lifetime of the protocol can be carried out up front. In fact, later in applying our protocols as the "inner protocol" of the final construction, we can use this reactive variant.

B Extension to Multi-Party Computation

In this section, we briefly sketch what is involved in extending our results to the multi-party case.

The protocol in [IPS08] extends to more than two parties, given inner and outer protocols for that many parties. The outer protocols from [DI06] and [CFIK03] do extend to the multi-party setting (called the "multi-client" setting in [IPS08] for more details). Hence by extending our inner protocol to the multi-party setting, all our results extend similarly.

In the general multi-party case the only non-trivial kind of computations carried out by the servers in the outer protocol is as follows:

- Each party P_i ($i = 1, \ldots, m$) sends x_i and y_i to the server.
- The server computes random values c_i such that $\Sigma_i c_i = (\Sigma_i x_i)(\Sigma_i y_i)$. Each party P_i is given c_i as the output.

A protocol for this using $\mathcal{F}_{\text{pdt-shr}}$ is as follows:

- For each ordered pair (i, j), $i \neq j$, parties P_i and P_j engage in an instance of $\mathcal{F}_{\text{pdt-shr}}$ with inputs (x_i, y_j) and obtain outputs $(\alpha_i^{(i,j)}, \alpha_j^{(i,j)})$, respectively, where $\alpha_i^{(i,j)} + \alpha_j^{(i,j)} = x_i y_j$.
- P_i outputs $x_i y_i + \Sigma_{j \neq i} \left(\alpha_i^{(i,j)} + \alpha_i^{(j,i)} \right)$.

The correctness of this protocol, and its (perfect) privacy against passive corruptions, is standard and analogous to the binary case from [GMW87].

C An Efficient Outer MPC Protocol

In this section, which is adapted from a preliminary full version of [IPS08], we describe a variant of the protocol from [DI06] which we use as the efficient outer MPC protocol in our constructions. We restrict the attention to the case of black-box fields (alternatively, pseudo-fields), and assume that the field size is super-polynomial in the security parameter. (This assumption can be removed at a minor cost to the arithmetic complexity.)

The protocol involves n servers and m clients ($m = 2$ by default), where only clients have inputs and outputs. The protocol is statistically UC-secure against an adaptive adversary corrupting an arbitrary number of clients and some constant fraction of the servers. We note that unlike the protocol from [DI06], here we do not need to guarantee output delivery and may settle for the weaker notion of "security with abort". This makes the protocol simpler, as it effectively means that whenever an inconsistency is detected by an honest party, this party can broadcast a complaint which makes the protocol abort.

For simplicity we assume that all $n + m$ parties in the MPC protocol have common access to an oracle which broadcasts random field elements, and do not count these elements towards the communication complexity. In [DI06] this is emulated via a distributed coin-flipping protocol and an ϵ-biased generator [NN90], which reduce the communication cost of implementing this procedure. Alternatively, random field elements can be directly generated by the m clients in the final protocol via efficient coin-flipping in the OT-hybrid model.

Before describing the protocol, we summarize its main efficiency features. For simplicity we shall restrict ourselves to $n = O(k)$, where k is a statistical security parameter, and a constant number of clients m. To evaluate an arithmetic circuit C of size s and multiplicative depth d, the arithmetic communication

complexity is $O(s + kd)$.[18] Assuming broadcast as an atomic primitive, the protocol requires $O(d)$ rounds. (We note that in the final m-party protocol obtained via the technique of [IPS08], broadcast only needs to be performed among the clients; in particular, in the two-party case broadcast can be implemented by directly sending the message.)

The computational complexity will be addressed after we describe the protocol.

To simplify the following exposition we will only consider the case of two clients Alice and Bob. An extension to the case of a larger number of clients is straightforward.

Another simplifying assumption is that the circuit C consists of d layers, where each layer performs addition, subtraction, or multiplication operations on values produced by the previous layer only. Circuits of an arbitrary structure can be easily handled at a worst-case additive cost of $O(nd)$, independently of the circuit size. (This cost can be amortized away for almost any natural instance of a big circuit. For instance, a sufficient condition for eliminating this cost is that for any two connected layers there are at least n wires connecting between the layers.)

C.1 Building Blocks

The protocol relies on tools and sub-protocols that we describe below.

Secret sharing for blocks. Shamir's secret sharing scheme [Sha79] distributes a secret $s \in F$ by picking a random degree-δ polynomial p such that $p(0) = s$, and sending to server j the point $p(j)$. Here F is a finite field such that $|F| > n$. By $1, 2, \ldots, n$ we denote distinct interpolation points, which in the case of a black-box access to F can be picked at random. The generalization of Franklin and Yung [FY92] achieves far better efficiency with a minor cost to the security level. In this scheme, a *block* of ℓ secrets $(s_1, \ldots, s_\ell)$ is shared by picking a random degree-δ polynomial p such that $p(1-j) = s_j$ for all j, and distributing to server j the point $p(j)$. (Here we assume that $-\ell + 1, \ldots, n$ denote $n + \ell$ distinct field elements.) Any set of $\delta + 1$ servers can recover the entire block of secrets by interpolation. On the other hand, any set of $t = \delta - \ell + 1$ servers learn nothing about the block of secrets from their shares. (Secret sharing schemes in which there is a gap between the privacy and reconstruction thresholds are often referred to as "ramp schemes".) For our purposes, we will choose ℓ to be a small constant fraction of n and δ a slightly bigger constant fraction of n (for instance, one can choose $\delta = n/3$ and $\ell = n/4$). This makes the amortized communication overhead of distributing a field element constant, while maintaining secrecy against a constant fraction of the servers.

Adding and multiplying blocks. Addition (or subtraction) and multiplication of shared blocks is analogous to the use of Shamir's scheme in the BGW protocol [BGW88]. Suppose that a block $a = (a_1, \ldots, a_\ell)$ was shared via a polynomial p_a and a block $b = (b_1, \ldots, b_\ell)$ was shared via a polynomial p_b. The servers can then locally compute shares of the polynomial $p_a + p_b$, which are valid shares for the sum $a + b$ of the two blocks. If each server multiplies its two local shares, the resulting n points are a valid secret-sharing using the degree-(2δ) polynomial $p = p_a p_b$ of the block $ab = (a_1 b_1, \ldots, a_\ell b_\ell)$. Note, however, that even if p_a, p_b were obtained from a random secret sharing, $p_a p_b$ is not a random degree-(2δ) secret sharing of ab. Thus, if we want to reveal ab we will need to mask $p_a p_b$ by a random degree-$2d$ secret-sharing of a block of 0's before revealing it. Also, in order to use ab for subsequent computations we will need to reduce its degree back to δ.

[18] While we do not attempt here to optimize the additive $O(kd)$ term, we note that a careful implementation of the protocol seems to make this term small enough for practical purposes. In particular, the dependence of this term on d can be eliminated for typical circuits.

Proving membership in a linear space. The protocol will often require a client to distribute to the servers a vector $v = (v_1, \ldots, v_n)$ (where each v_j includes one or more field elements) while assuring them that v belongs to some linear space L. This should be done while ensuring that the adversary does not learn more information about v than it is entitled to, and while ensuring the honest parties that the shares they end up with are consistent with L. For efficiency reasons, we settle for having the shares of the honest parties *close* to being consistent with L. Since we will only use this procedure with L that form an error correcting code whose minimal distance is a large constant multiple of δ, the effect of few "incorrect" shares can be undone via error-correction. (In fact, in our setting of security with abort error detection will be sufficient.) More concretely, our procedure takes input $v = (v_1, \ldots, v_n) \in L$ from a dealer D (Alice or Bob). In the presence of an active, adaptive adversary who may corrupt any client and at most t servers, it should have the following properties:

- Completeness: If D is uncorrupted then every honest server j outputs v_j.
- Soundness: Suppose D is corrupted. Except with negligible probability, either all honest servers reject (in which case the dealer is identified as being a cheater), or alternatively the joint outputs of all n servers are most $2t$-far (in Hamming distance) from some vector in $v \in L$.
- Zero-Knowledge: If D is uncorrupted, the adversary's view can be simulated from the shares v_j of corrupted servers.

Verifiable Secret Sharing (VSS) can be obtained by applying the above procedure on the linear space defined by the valid share vectors. Note that in contrast to standard VSS, we tolerate some inconsistencies to the shares on honest servers. Such inconsistencies will be handled by the robustness of the higher level protocol.

Implementing proofs of membership. We will employ a sub-protocol from [DI06] (Protocol 5.1) for implementing the above primitive. This protocol amortizes the cost of proving that many vectors $v^1, \ldots, v^q$ owned by the same dealer D belong to the same linear space L by taking random linear combinations of these vectors together with random vectors from L that are used for blinding. The high level structure of this protocol is as follows.

- *Distributing shares.* D distributes $v^1, \ldots, v^q$ to the servers.
- *Distributing blinding vectors.* D distributes a random vector $r \in L$ that is used for blinding. (This step ensures the zero-knowledge property; soundness does not depend on the valid choice of this r.)
- *Coin-flipping.* The players invoke the random field element oracle to obtain a length-q vector defining a random linear combination of the q vectors distributed by the dealer. (In [DI06] this is implemented using distributed coin-flipping and an ϵ-biased generator; in our setting this can be implemented directly by the clients in the OT-hybrid model. Moreover, in the case of two clients we let the other client, who does not serve as a dealer, pick r on its own.)
- *Proving.* The dealer computes the linear combination of its vectors v^i defined by r, and adds to it the corresponding blinding vector. It broadcasts the results.
- *Complaining.* Each server applies the linear combination specified by r to its part of the vectors distributed by the dealer, and ensures that the result is consistent with the value broadcast in the previous step. If any inconsistency is detected, the server broadcasts a complaint and the protocol aborts. Also, the protocol aborts if the vector broadcasted by the dealer is not in L.

- *Outputs.* If no server broadcasted a complaint, the servers output the shares distributed by the dealer in the first step (discarding the blinding vectors and the results of the coin-flips).

In the case of a static corruption of servers, if the shares dealt to honest servers are inconsistent, the protocol will abort except with $1/|F|$ probability, which is assumed to be negligible in k. The adaptive case is a bit more involved, since the adversary can choose which servers to corrupt only after the random linear combination is revealed. This case is easy to analyze via a union bound (which requires that $|F| > \binom{n}{t}$). Alternatively, a tighter analysis shows that if $|F|$ is superpolynomial in k then, except with negligible probability, either the protocol aborts or there exists a small set B of servers such that all shares held by the honest servers *excluding* those in B are consistent with a valid codeword from L. This condition is sufficient for the security of the protocol.

We will sometimes employ the above protocol in a scenario where vectors $v^1, \ldots, v^q$ are already distributed between the servers and known to the dealer, and the dealer wants to convince the servers that these shares are consistent with L. In such cases we will employ the above sub-protocol without the first step.

Proving global linear constraints. We will often need to deal with a more general situation of proving that vectors $v^1, \ldots, v^q$ not only lie in the same space L, but also satisfy additional global constraints. A typical scenario applies to the case where the v^i are shared blocks defined by degree-δ polynomials. In such a case, we will need to prove that the secrets shared in these blocks satisfy a specified replication pattern (dictated by the structure of the circuit C we want to compute). Such a replication pattern specifies which entries in the q blocks should be equal. An observation made in [DI06] is that: (1) such a global constraint can be broken into at most $q\ell$ atomic conditions of the type "entry i in block j should be equal to entry i' in block j'", and (2) by grouping these atomic conditions into ℓ^2 types defined by (i, i'), we can apply the previous verification procedure to simultaneously verify all conditions in the same type. That is, to verify all conditions of type (i, i') each server concatenates his two shares of every pair of blocks that should be compared in this type, and then applies the previous verification procedure with L being the linear space of points on degree-δ polynomials (p_1, p_2) which satisfy the constraint $p_1(1 - i) = p_2(1 - i')$. Unlike [DI06] we will also employ the above procedure in the case where p_1, p_2 may be polynomials of different degrees (e.g., δ and 2δ), but the same technique applies to this more general case as well.

C.2 The Protocol

The protocol is a natural extension of the protocol from [DI06], which can be viewed as handling the special case of constant-depth circuits using a constant number of rounds. We handle circuits of depth d by using $O(d)$ rounds of interaction. The protocol from [DI06] handles general functions by first encoding them into $\mathbf{NC}^0$ functions, but such an encoding step is too expensive for our purposes and in any case does not apply to the arithmetic setting. The protocol is simplified by the fact that we only need to achieve "security with abort", as opposed to the full security of the protocol from [DI06].

Recall that we assume the circuit C to consist of d layers each, and that each gate in layer i depends on two outputs from from layer $i - 1$.

The high level strategy is to pack the inputs for each layer into blocks in a way that allows to evaluate multiplication, addition, and subtraction gates in this layer "in parallel" on pairs of blocks. That is, the computation of the layer will consist of disjoint parallel computations of the form $a \cdot b$, $a + b$, and $a - b$, where a and b are blocks of ℓ binary values and the ring operation is performed coordinate-wise. This will require blocks to be set up so that certain inputs appear in several places. Such a replication pattern will be

enforced using the procedure described above. Throughout the protocol, if a prover is caught cheating the protocol is aborted.

The protocol will proceed as follows:

1. *Sharing inputs.* The clients arrange their inputs into blocks with a replication pattern that sets up the parallel evaluation for the first layer (namely, so that the first layer will be evaluated by applying the same arithmetic operation to blocks 1,2, to blocks 3,4, etc.). Each client then secret-shares its blocks, proving to the servers that the shares of each block agree with a polynomial of degree at most δ and that the secrets in the shared blocks satisfy the replication pattern determined by the first layer of C. (Such proofs are described in the previous section.)

   If we want to enforce input values to be boolean (namely, either 0 or 1) this can be done a standard way by letting the servers securely reveal $1 - a \cdot a$ for each block a (which should evaluate to a block of 0's).

2. *Evaluating C on shared blocks.* The main part of the protocol is divided into d phases, one for evaluating each layer of C. For $h = 1, 2, \ldots, d$ we repeat the following:

   - *Combining and and blinding.* At the beginning of the phase, the inputs to layer h are arranged into blocks, so that the outputs of layer h can be obtained by performing some arithmetic operation on each consecutive pair of blocks. Moreover, each block is secret-shared using a degree-δ polynomial. Addition and subtraction on blocks can be handled non-interactively by simply letting each server locally add or subtract its two shares. In the following we address the more involved case of multiplication. We would like to reveal the outputs of the layer to Alice, masked by random blinding blocks picked by Bob. For this, Bob will VSS random blocks, one for each block of output. The secret-sharing of these blocks is done using polynomials of degree 2δ.

     (Again, verifying that the shares distributed by Bob are valid is done using the procedure described above.) For every pair of input blocks a, b whose product is computed, each server j locally computes the degree-2 function $c(j) = a(j)b(j) + r(j)$, where $a(j), b(j)$ are its shares of a, b and $r(j)$ is its share of the corresponding blinding block r distributed by Bob. For each pair of blocks combined in this way, the server sends his output (a single field element) to Alice. Note that the points $c(j)$ lie on a random degree-2δ polynomial p_c, and thus reveal no information about a, b. Moreover, the polynomial p_c can be viewed as some valid degree-2δ secret sharing of the block $c = ab + r$.

   - *Reducing degree and rearranging blocks for layer $h+1$.* Alice checks that the points $c(j)$ indeed lie on a polynomial p_c of degree at most 2δ (otherwise she aborts). Then she recovers the blinded output block $c = ab + r$ by letting $c_j = p_c(1 - j)$. Now Alice uses all blinded blocks c obtained in this way to set up the (blinded) blocks for computing layer $h + 1$.

     For this, she sets up a new set of blocks that are obtained by applying a projection (namely, permuting and copying) to the blocks c that corresponds to the structure of layer $h + 1$. (In particular, the number of new blocks in which an entry in a block c will appear is precisely the fan-out of the corresponding wire in C.) Let c' denote the rearranged blinded blocks.

     Now Alice secret-shares each block c' using a degree-δ polynomial $p_{c'}$. She needs to prove to the servers that the shares she distributed are of degree δ and that the entries of the shared blocks c' satisfy the required relation with respect to the blocks c that are already shared between the servers using degree-2δ polynomials. Such a proof can be efficiently carried out using the procedure described above. Note that pairs of polynomials $(p_c, p_{c'})$ such that p_c is of degree

at most 2δ, $p_{c'}$ is of degree at most δ, and $p_c(i) = p_{c'}(j)$ form a linear space (for any fixed i, j), and hence the $2n$ evaluations of such polynomials on the points that correspond to the servers form a linear subspace of F^{2n}. Also, the corresponding code will have a large minimal distance because of the degree restriction, which ensures that the adversary cannot corrupt a valid codeword without being detected (or even corrected, in the setting of security without abort).

- *Unblinding.* To set up the input blocks for the evaluation of layer $h + 1$, we need to cancel the effect of the blinding polynomials p_r distributed by Bob. For this, Bob distributes random degree-δ unblinding polynomials $p_{r'}$ that encode blocks r' obtained by applying to the r blocks the same projection defined by the structure of layer $h+1$ that was applied by Alice. Bob proves that the polynomials $p_{r'}$ are consistent with the p_r similarly to the corresponding proof of Alice in the previous step. (In fact, both sharing the $p_{r'}$ and proving their correctness could be done in the first step.) Finally, each server obtains its share of an input block a for layer $h + 1$ by letting $a(j) = c'(j) - r'(j)$.

3. *Delivering outputs.* The outputs of C are revealed to the clients by having the servers send their shares of each output block to the client who should receive it. The client checks that the n values received for each block are consistent with a degree-d polynomial (otherwise it aborts), and recovers the output of this block.

Communication complexity. By the choice of parameters, the communication overhead of encoding each block of field elements is constant. Accounting for narrow layers (whose size is smaller than one block) as well as wires between non-adjacent layers, we get an additive arithmetic communication overhead of $O(nd)$ (accounting for the worst-case scenario of one may need to maintain a block of values to be used in each subsequent layer). As noted above, this overhead can be reduced or even eliminated in most typical cases. Finally, the cost of picking random field elements for the random linear combinations can be reduced via the use of (arithmetic) ϵ-biased generators or directly improved via an alternative procedure described below.

Computational complexity. Using known FFT-based techniques for multipoint polynomial evaluation and interpolation, both the secret sharing and the reconstruction of a block of length ℓ with $n = O(\ell)$ servers can be done with arithmetic complexity of $O(\ell \log^2 \ell)$ [vzGG99]. Choosing evaluation points which are n-th roots of unity, this complexity can be reduced to $O(\ell \log \ell)$ (at the expense of sacrificing the black-box use of the field). The computational bottleneck in the above protocol is the procedure for verifying that shared blocks satisfy the replication pattern corresponding to C. This can be improved by converting C into an equivalent circuit C' which reduces the overhead of this procedure. A more direct and efficient way for implementing the above procedure can be obtained by adapting an idea from [Gro08] to our setting. To test that a set of M blocks v_i satisfies a given replication pattern, pick a set of M random blocks r_i and test that $\sum v_i r_i = \sum v_i r'_i$, where the blocks r'_i are obtained by permuting the blocks in r_i along the "cycles" defined by the replication pattern. (That is, for each set of positions in the blocks v_i which should be tested to be equal, apply a cyclic shift to the values in the corresponding entries of the blocks r_i.) This sum of inner products can be computed by adding up all pointwise-products of v_i and r'_i together with a random block whose entries add up to 0.