Between a Rock and a Hard Place: Interpolating Between MPC and FHE

A. Choudhury, J. Loftus, E. Orsini, A. Patra and N. P. Smart

Dept. Computer Science,
University of Bristol,
United Kingdom.
{Ashish.Choudhary,Emmanuela.Orsini,Arpita.Patra}@bristol.ac.uk,
{loftus,nigel}@cs.bris.ac.uk

Abstract. We present a computationally secure MPC protocol for threshold adversaries which is parametrized by a value L. When $L = 2$ we obtain a classical form of MPC protocol in which interaction is required for multiplications, as L increases interaction is reduced, in that one requires interaction only after computing a higher degree function. When L approaches infinity one obtains the FHE based protocol of Gentry, which requires no interaction. Thus one can trade communication for computation in a simple way. Our protocol is based on an interactive protocol for "bootstrapping" a somewhat homomorphic encryption (SHE) scheme. The key contribution is that our presented protocol is highly communication efficient enabling us to obtain reduced communication when compared to traditional MPC protocols for relatively small values of L.

1 Introduction

In the last few years computing on encrypted data via either Fully Homomorphic Encryption (FHE) or Multi-Party Computation (MPC) has been subject to a remarkable number of improvements. Firstly, FHE was shown to be possible [23]; and this was quickly followed by a variety of applications and performance improvements [6, 9, 8, 24, 25, 29, 30]. Secondly, whilst MPC has been around for over thirty years, only in the last few years we have seen an increased emphasis on practical instantiations; with some very impressive results [5, 18, 28].

We focus on MPC where n parties wish to compute a function on their respective inputs. Whilst the computational overhead of MPC protocols, compared to computing "in the clear", is relatively small (for example in practical protocols such as [20, 28] a small constant multiple of the "in the clear" cost), the main restriction on practical deployment of MPC is the communication cost. Even for protocols in the preprocessing model, evaluating arithmetic circuits over a field $\mathbb{F}_p$, the communication cost in terms of number of bits per multiplication gate and per party is a constant multiple of the bit length, $\log p$, of the data being manipulated for a typically large value of the constant. This is a major drawback of MPC protocols since communication is generally more expensive than computation. Theoretical results like [15] (for the computational case) and [16] (for the information theoretic case) bring down the per gate per party communication cost to a very small quantity; essentially $\mathcal{O}(\frac{\log n}{n} \cdot \log |C| \cdot \log p)$ bits for a circuit C of size $|C|$. While these results suggest that the communication cost can be asymptotically brought down to a constant for large n, the constants are known to be large for any practical purpose. Our interest lies in constructing efficient MPC protocols where the efficiency is measured in terms of *exact* complexity rather than the *asymptotic* complexity.

In his thesis, Gentry [22] showed how FHE can be used to reduce the communication cost of MPC down to virtually zero for any number of parties. In Gentry's MPC protocol all parties send to each other the encryptions of their inputs under a shared FHE public key. They then compute the function homomorphically, and at the end perform a shared decryption. This implies an MPC protocol whose communication is limited to a function of the input and output sizes, and not to the complexity of the circuit. However, this reduction in communication complexity comes at a cost, namely the huge expense of evaluating homomorphically the function. With current understanding of FHE technology, this solution is completely infeasible in practice.

A variant of Gentry's protocol was presented by Asharov et al. in [1] where the parties outsource their computation to a server and only interact via a distributed decryption. The key innovation in [1] was that independently generated (FHE) keys can be combined into a "global" FHE key with distributed decryption capability. We do not assume such a functionality of the keys (but one can easily extend our results to accommodate this); instead we focus on using distributed decryption to enable *efficient* multi-party bootstrapping. In addition the work of [1], in requiring an FHE scheme, as opposed to the somewhat homomorphic encryption (SHE) scheme of our work, requires the assumption of circular security of the underlying FHE scheme (and hence more assumptions). Although in our instantiation, for efficiency reasons, we use a scheme which assumes circular security; this is not however theoretically necessary.

In [20], following on the work in [4], the authors propose an MPC protocol which uses an SHE scheme as an "optimization". Based in the preprocessing model, the authors utilize an SHE scheme which can evaluate circuits of multiplicative depth one to optimize the preprocessing step of an essentially standard MPC protocol. The optimizations, and use of SHE, in [20] are focused on the case of computational improvements. In this work we invert the use of SHE in [20], by using it for the online phase of the MPC protocol, so as to optimize the communication efficiency for any number of parties.

In essence we interpolate between the two extremes of traditional MPC protocols (with high communication but low computational costs) and Gentry's FHE based solution (with high computation but low communication costs). Our interpolation is dependent on a parameter, which we label as L, where $L \geq 2$. At one extreme, for $L = 2$ our protocol resembles traditional MPC protocols, whilst at the other extreme, for $L = \infty$ our protocol is exactly that of Gentry's FHE based solution. We emphasize that our construction is general in that *any* SHE can be used which supports homomorphic computation of depth *two* circuits and threshold decryption. Thus the requirements on the underlying SHE scheme are much weaker than the previous SHE (FHE) based MPC protocols, such as the one by Asharov et al. [1], which relies on the specifics of LWE (learning with errors) based SHE i.e. *key-homomorphism* and demands homomorphic computation of depth L circuits for big enough L to bootstrap.

The solution we present is in the preprocessing model; in which we allow a preprocessing phase which can compute data which is neither input, nor function, dependent. This preprocessed data is then consumed in the online phase. As usual in such a model our goal is for efficiency in the online phase only. We present our basic protocol and efficiency analysis for the case of passive threshold adversaries only; i.e. we can tolerate up to t passive corruptions where $t < n$. We then note that security against t active adversaries with $t < n/3$ can be achieved for no extra cost in the online phase. For the active security case, essentially the same communication costs can be achieved even when $t < n/2$, bar some extra work (which is *independent* of $|C|$) to eliminate the cheating parties when they are detected. The security of our protocols are proven in the standard universal composability (UC) framework [10].

Finally we note that our results on communication complexity, both in a practical and in an asymptotic sense, in the computational setting are comparable (if not better) than the best known results in the information theoretic and computational settings. Namely the best known optimally resilient statistically secure MPC protocol with $t < n/2$ has (asymptotic) communication complexity of $\mathcal{O}(n)$ per multiplication [3], whereas ours is $\mathcal{O}(n/L)$ (see Section 6 for the analysis of our protocol). With near optimal resiliency of $t < (\frac{1}{3} - \epsilon)n$, the best known perfectly secure MPC protocol has (asymptotic) communication complexity of $\mathcal{O}(\text{polylog } n)$ per multiplication [16], but a huge constant is hiding under the $\mathcal{O}$. In the computational settings, with near optimal resiliency of $t < (\frac{1}{2} - \epsilon)n$, the best known MPC protocol has (asymptotic) communication complexity of $\mathcal{O}(\text{polylog } n)$ per multiplication [15], but again a huge constant is hiding under the $\mathcal{O}$. All these protocols can not win over ours when *exact* communication complexity is compared for even small values of L.

Overview: Our protocol is intuitively simple. We first take an L-levelled SHE scheme (strictly it has $L + 1$ levels, but can evaluate circuits with L levels of multiplications) which possesses a distributed decryption protocol for the specific access structure required by our MPC protocol. We assume that the SHE scheme is implemented over a ring which supports N embeddings of the underlying finite field $\mathbb{F}_p$ into the message space of the SHE scheme. Almost all known SHE schemes support such packing of the finite field into the plaintext slots in an SIMD manner [24, 30]; and such packing has been crucial in the implementation of SHE in various applications [17, 20, 25].

Clearly with such a setup we can implement Gentry's MPC solution for circuits of multiplicative depth L. All that remains is how to "bootstrap" from circuits with multiplicative depth L to arbitrary circuits. The standard solution would be to bootstrap the FHE scheme directly, following the blueprint outlined in Gentry's thesis. However, in the case of applications to MPC we could instead utilize a protocol to perform the bootstrapping. In a nutshell that is exactly what we propose.

The main issue then is show how to efficiently perform the bootstrapping in a distributed manner; where efficiency is measured in terms of computational and communication performance. Naively performing an MPC protocol to execute the bootstrapping phase will lead to a large communication overhead, due to the inherent overhead in dealing with homomorphic encryptions. But on its own this is enough to obtain our asymptotic interpolation between FHE and MPC; we however aim to provide an efficient and practical interpolation. That is one which is efficient for small values of L. It turns out that a special case of a suitable bootstrapping protocol can be found as a sub-procedure of the MPC protocol in [20]. We extract the required protocol, generalise it, and then apply it to our MPC situation.

To ease exposition we will not utilize the packing from [24] to perform evaluations of the depth L sub-circuits; we see this as a computational optimization which is orthogonal to the issues we will explore in this paper. In any practical instantiation of the protocol of this paper such a packing could be used, as described in [24], in evaluating the circuit of multiplicative depth L. However, we will use this packing to perform the bootstrapping in a communication efficient manner.

The bootstrapping protocol runs in two phases. In the first (offline) phase we repeatedly generate sets of ciphertexts, one set for each party, such that all parties learn the ciphertexts but only the given party learns their underlying messages (which are assumed to be packed). The offline phase can be run in either a passive, covert or active security model, irrespective of the underlying access structure of the MPC protocol following ideas from [18]. In the second (online) phase the data to be bootstrapped is packed together, a random mask is added (computed from the offline phase data), a distributed decryption protocol is executed to obtain the masked data which is then re-encrypted, the mask is subtracted and then the data is unpacked. All these steps are relatively efficient, with communication only being required for the distributed decryption.

To apply our interactive bootstrapping method efficiently we need to make a mild assumption on the circuit being evaluated; this is similar to the assumptions used in [15, 16, 21]. The assumption can be intuitively seen as saying that the circuit is relatively wide enough to enable packing of enough values which need to be bootstrapped at each respective level. We expect that most circuits in practice will satisfy our assumption, and we will call the circuits which satisfy our requirement "well formed".

We pause to note that the ability to open data within the MPC protocol enables one to perform more than a simple evaluation of an arithmetic circuit. This observation is well known in the MPC community, where it has been used to obtain efficient protocols for higher level functions [11, 14]. Thus enabling a distributed bootstrapping also enables one to produce more efficient protocols than purely FHE based ones.

We instantiate our protocol with the BGV scheme [7] and obtain sufficient parameter sizes following the methodology in [18, 25]. Due to the way we utilize the BGV scheme we need to restrict to MPC protocols for arithmetic circuits over a finite field $\mathbb{F}_p$, with $p \equiv 1 \pmod{m}$ with $m = 2 \cdot N$ and $N = 2^r$ for some r. The distributed decryption method uses a "smudging" technique (see the full version of the paper) which means that the modulus used in the BGV scheme needs to be larger than what one would need to perform just the homomorphic operations. Removing this smudging technique, and hence obtaining an efficient protocol for distributed decryption, for *any* SHE scheme is an interesting open problem; with many potential applications including that described in this paper.

We show that even for a very small value of L, in particular $L = 5$, we can achieve better communication efficiency than many practical MPC protocols in the preprocessing model. Most practical MPC protocols such as [5, 20, 28] require the transmission of at least two finite field elements per multiplication gate between each pair of parties. In

[20] a technique is presented which can reduce this to the transmission of an average of three field elements per multiplication gate per party (and not per pair of parties). Note the models in [5] (three party, one passive adversary) and [20, 28] (n party, dishonest majority, active security) are different from ours (we assume honest majority, active security); but even mapping these protocols to our setting of n party honest majority would result in the same communication characteristics. We show that for relatively small values of L, i.e. $L > 8$, one can obtain a communication efficiency of less than one field element per gate and party (details available in Section 6).

Clearly, by setting L appropriately one can obtain a communication efficiency which improves upon that in [15, 16]; albeit we are only interested in communication in the online phase of a protocol in the preprocessing model whilst [15, 16] discuss total communication cost over all phases. But we stress this is not in itself interesting, as Gentry's FHE based protocol can beat the communication efficiency of [15, 16] in any case. What is interesting is that we can beat the communication efficiency of the online phase of practical MPC protocols, with very small values of L indeed. Thus the protocol in this paper may provide a practical tradeoff between existing MPC protocols (which consume high bandwidth) and FHE based protocols (which require huge computation).

Our protocol therefore enables the following use-case: it is known that SHE schemes only become prohibitively computationally expensive for large L; indeed one of the reasons why the protocols in [18, 20] are so efficient is that they restrict to evaluating homomorphically circuits of multiplicative depth one. With our protocol parties can a priori decide the value of L, for a value which enables them to produce a computationally efficient SHE scheme. Then they can execute an MPC protocol with communication costs reduced by effectively a factor of L. Over time as SHE technology improves the value of L can be increased and we can obtain Gentry's original protocol. Thus our methodology enables us to interpolate between the case of standard MPC and the eventual goal of MPC with almost zero communication costs.

2 Well Formed Circuits

In this section we define what we mean by well formed circuits, and the pre-processing which we require on our circuits. We take as given an arithmetic circuit C defined over a finite field $\mathbb{F}_p$. In particular the circuit C is a directed acyclic graph consisting of edges made up of n_I input wires, n_O output wires, and n_W internal wires, plus a set of nodes being given by a set of gates $\mathbb{G}$. The gates are divided into sets of Add gates $\mathbb{G}_A$ and Mult gates $\mathbb{G}_M$, with $\mathbb{G} = \mathbb{G}_A \cup \mathbb{G}_M$, with each $\mathsf{Add}/\mathsf{Mult}$ gate taking two wires (or a constant value in $\mathbb{F}_p$) as input and producing one wire as output. The circuit is such that all input wires are open on their input ends, and all output wires are open on their output ends, with the internal wires being connected on both ends. We let the depth of the circuit d be the length of the maximum path from an input wire to an output wire. Our definition of a well formed circuit is parametrized by two positive integer values N and L.

We now associate inductively to each wire in the circuit an integer valued label as follows. The input wires are given the label one; then all other wires are given a label as follows (where we assume a constant input to a gate has label L):

$$\begin{aligned}\text{Label of output wire of } \mathsf{Add} \text{ gate } &= \min(\text{Label of input wires}),\\ \text{Label of output wire of } \mathsf{Mult} \text{ gate } &= \min(\text{Label of input wires}) - 1.\end{aligned}$$

Thus the minimum value of a label is $1 - d$ (which is negative for a general d). Looking ahead, the reason for starting with an input label of one is when we match this up with our MPC protocol this will result in low communication complexity for the input stage of the computation.

We now augment the circuit, to produce a new circuit C^{aug} which will have labels in the range $[1, \ldots, L]$, by adding in some special gates which we will call Refresh gates; the set of such gates are denoted as $\mathbb{G}_R$. A Refresh gate takes as input a maximum of N wires, and produces as output an exact copy of the specified input wires. The input requirement is that the input wires must have label in the range $[1, \ldots, L]$, and all that the Refresh gate does is relabel the labels of the gate's input wires to be L. At the end of the augmentation process we require the invariant that all wire labels in C^{aug} are then in the range $[1, \ldots, L]$, and the circuit is now essentially a collection of "sub-circuits" of multiplicative depth at most $L - 1$ glued together using Refresh gates. However, we require that this is done with as small a number of Refresh gates as possible.

Definition 1 (Well Formed Circuit). *A circuit C will be called well formed if the number of* Refresh *gates in the associated augmented circuit C^{aug} is at most $\frac{2 \cdot |\mathbb{G}_M|}{L \cdot N}$.*

We expect that "most" circuits will be well formed due to the following argument: We first note that the only gates which concern us are multiplication gates; so without loss of generality we consider a circuit C consisting only of multiplication gates. The circuit has d layers, and let the width of C (i.e. the number of gates) at layer i be w_i. Consider the algorithm to produce C^{aug} which considers each layer in turn, from $i = 1$ to d and adds Refresh gates where needed. When reaching level i in our algorithm to produce C^{aug} we can therefore assume (by induction) that all input wires at this layer have labels in the range $[1, \ldots, L]$. To maintain the invariant we only need to apply a Refresh operation to those input wires which have label one. Let p_i denote the proportion of wires at layer i which have label one when we perform this process. It is clear that the number of required Refresh gates which we will add into C^{aug} at level i will be at most $\lceil 2 \cdot p_i \cdot w_i / N \rceil$, where the factor of two comes from the fact that each multiplication gate has two input wires.

Assuming a large enough circuit we can assume for most layers that this proportion p_i will be approximately $1/L$, since wires will be refreshed after their values have passed through L multiplication gates. So summing up over all levels, the expected number of Refresh gates in C^{aug} will be:

$$\sum_{i=1}^{d} \left\lceil \frac{2 \cdot w_i}{L \cdot N} \right\rceil \approx \frac{2}{L \cdot N} \cdot \sum_{i=1}^{d} w_i = \frac{2 \cdot |\mathbb{G}_M|}{L \cdot N}.$$

Note, we would expect that for most circuits this upper bound on the number of Refresh gates could be easily met. For example our above rough analysis did not take into account the presence of gates with fan-out greater than one (meaning there are less wires to Refresh than we estimated above), nor did it take into account utilizing unused slots in the Refresh gates to refresh wires with labels not equal to one.

Determining an optimum algorithm for moving from C to a suitable C^{aug}, with a minimal number of Refresh gates, is an interesting optimization problem which we leave as an open problem; however clearly the above outlined greedy algorithm will work for most circuits.

3 Threshold L-Levelled Packed Somewhat Homomorphic Encryption (SHE)

In this section, we present a detailed explanation of the syntax and requirements for our Threshold L-Levelled Packed Somewhat Homomorphic Encryption Scheme. The scheme will be parametrized by a number of values; namely the security parameter κ, the number of levels L, the amount of packing of plaintext elements which can be made into one ciphertext N, a statistical security parameter sec (for the security of the distributed decryption) and a pair (t, n) which defines the threshold properties of our scheme. In practice the parameter N will be a function of L and κ. The message space of the SHE scheme is defined to be $\mathcal{M} = \mathbb{F}_p^N$, and we embed the finite field $\mathbb{F}_p$ into $\mathcal{M}$ via a map $\chi : \mathbb{F}_p \longrightarrow \mathcal{M}$.

Let $\mathcal{C}(L)$ denote the family of circuits consisting of addition and multiplication gates whose labels follow the conventions in Section 2; except that input wires have label L and whose minimum wire label is zero. Thus $\mathcal{C}(L)$ is the family of standard arithmetic circuits of multiplicative depth at most L which consist of 2-input addition and multiplication gates over $\mathbb{F}_p$, whose wire labels lie in the range $[0, \ldots, L]$. Informally, a threshold L-levelled SHE scheme supports homomorphic evaluation of any circuit in the family $\mathcal{C}(L)$ with the provision for distributed (threshold) decryption, where the input wire values v_i are mapped to ciphertexts (at level L) by encrypting $\chi(v_i)$.

As remarked in the introduction we could also, as in [24], extend the circuit family $\mathcal{C}(L)$ to include gates which process N input values at once as

$$N\text{-}\mathsf{Add}\left(\langle u_1, \ldots, u_N\rangle, \langle v_1, \ldots, v_N\rangle\right) := \langle u_1 + v_1, \ldots, u_N + v_N\rangle,$$
$$N\text{-}\mathsf{Mult}\left(\langle u_1, \ldots, u_N\rangle, \langle v_1, \ldots, v_N\rangle\right) := \langle u_1 \times v_1, \ldots, u_N \times v_N\rangle.$$

But such an optimization of the underlying circuit is orthogonal to our consideration. However, the underlying L-levelled packed SHE scheme supports such operations on its underlying plaintext (we will just not consider these operations in our circuits being evaluated).

We can evaluate subcircuits in $\mathcal{C}(L)$; and this is how we will describe the homomorphic evaluation below (this will later help us to argue the correctness property of our general MPC protocol). In particular if $C \in \mathcal{C}(L)$, we can deal with sub-circuits C^{sub} of C whose input wires have labels $\mathfrak{l}_1^{in}, \ldots, \mathfrak{l}_{\ell_{in}}^{in}$, and whose output wires have labels $\mathfrak{l}_1^{out}, \ldots, \mathfrak{l}_{\ell_{out}}^{out}$, where $\mathfrak{l}_i^{in}, \mathfrak{l}_i^{out} \in [0, \ldots, L]$. Then given ciphertexts $\mathfrak{c}_1, \ldots, \mathfrak{c}_{\ell_{in}}$ encrypting the messages $\mathbf{m}_1, \ldots, \mathbf{m}_{\ell_{in}}$, for which the ciphertexts are at level $\mathfrak{l}_1^{in}, \ldots, \mathfrak{l}_{\ell_{in}}^{in}$, the homomorphic evaluation function will produce ciphertexts $\hat{\mathfrak{c}}_1, \ldots, \hat{\mathfrak{c}}_{\ell_{out}}$, at levels $\mathfrak{l}_1^{out}, \ldots, \mathfrak{l}_{\ell_{out}}^{out}$, which encrypt the messages corresponding to evaluating C^{sub} on the components of the vectors $\mathbf{m}_1, \ldots, \mathbf{m}_{\ell_{in}}$ in a SIMD manner. More formally:

Definition 2 (Threshold L-levelled Packed SHE). *An L-levelled public key packed somewhat homomorphic encryption (SHE) scheme with the underlying message space $\mathcal{M} = \mathbb{F}_p^N$, public key space $\mathcal{PK}$, secret key space $\mathcal{SK}$, evaluation key space $\mathcal{EK}$, ciphertext space $\mathcal{CT}$ and distributed decryption key space $\mathcal{DK}_i$ for $i \in [1, \ldots, n]$ is a collection of the following PPT algorithms, parametrized by a computational security parameter κ and a statistical security parameter* sec*:*

1. $\mathsf{SHE.KeyGen}(1^\kappa, 1^{\mathsf{sec}}, n, t) \to (\mathsf{pk}, \mathsf{ek}, \mathsf{sk}, \mathsf{dk}_1, \ldots, \mathsf{dk}_n)$*: The key generation algorithm outputs a public key* $\mathsf{pk} \in \mathcal{PK}$*, a public evaluation key* $\mathsf{ek} \in \mathcal{EK}$*, a secret key* $\mathsf{sk} \in \mathcal{SK}$ *and n keys* $(\mathsf{dk}_1, \ldots, \mathsf{dk}_n)$ *for the distributed decryption, with* $\mathsf{dk}_i \in \mathcal{DK}_i$*.*
2. $\mathsf{SHE.Enc}_{\mathsf{pk}}(\mathbf{m}, r) \to (\mathfrak{c}, L)$*: The encryption algorithm computes a ciphertext $\mathfrak{c} \in \mathcal{CT}$, which encrypts a plaintext vector $\mathbf{m} \in \mathcal{M}$ under the public key* pk *using the randomness*[1] *r and outputs $(\mathfrak{c}, L)$ to indicate that the* associated level *of the ciphertext is L.*
3. $\mathsf{SHE.Dec}_{\mathsf{sk}}(\mathfrak{c}, \mathfrak{l}) \to \mathbf{m}'$*: The decryption algorithm decrypts a ciphertext $\mathfrak{c} \in \mathcal{CT}$ of associated level $\mathfrak{l}$ where $\mathfrak{l} \in [0, \ldots, L]$ using the decryption key* sk *and outputs a plaintext $\mathbf{m}' \in \mathcal{M}$. We say that $\mathbf{m}'$ is the* plaintext associated with $\mathfrak{c}$.
4. $\mathsf{SHE.ShareDec}_{\mathsf{dk}_i}(\mathfrak{c}, \mathfrak{l}) \to \bar{\mu}_i$*: The share decryption algorithm takes a ciphertext $\mathfrak{c}$ with associated level $\mathfrak{l} \in [0, \ldots, L]$, a key dk_i for the distributed decryption, and computes a decryption share $\bar{\mu}_i$ of $\mathfrak{c}$.*
5. $\mathsf{SHE.ShareCombine}((\mathfrak{c}, \mathfrak{l}), \{\bar{\mu}_i\}_{i \in [1, \ldots, n]}) \to \mathbf{m}'$*: The share combine algorithm takes a ciphertext $\mathfrak{c}$ with associated level $\mathfrak{l} \in [0, \ldots, L]$ and a set of n decryption shares and outputs a plaintext $\mathbf{m}' \in \mathcal{M}$.*
6. $\mathsf{SHE.Eval}_{\mathsf{ek}}(C^{\mathsf{sub}}, (\mathfrak{c}_1, \mathfrak{l}_1^{in}), \ldots, (\mathfrak{c}_{\ell_{in}}, \mathfrak{l}_{\ell_{in}}^{in})) \to (\hat{\mathfrak{c}}_1, \mathfrak{l}_1^{out}), \ldots, (\hat{\mathfrak{c}}_{\ell_{out}}, \mathfrak{l}_{\ell_{out}}^{out})$*: The homomorphic evaluation algorithm is a deterministic polynomial time algorithm (polynomial in L, ℓ_{in}, ℓ_{out} and κ) that takes as input the evaluation key* ek*, a subcircuit C^{sub} of a circuit $C \in \mathcal{C}(L)$ with ℓ_{in} input gates and ℓ_{out} output gates as well as a set of ℓ_{in} ciphertexts $\mathfrak{c}_1, \ldots, \mathfrak{c}_{\ell_{in}}$, with associated level $\mathfrak{l}_1^{in}, \ldots, \mathfrak{l}_{\ell_{in}}^{in}$, and outputs ℓ_{out} ciphertexts $\hat{\mathfrak{c}}_1, \ldots, \hat{\mathfrak{c}}_{\ell_{out}}$, with associated levels $\mathfrak{l}_1^{out}, \ldots, \mathfrak{l}_{\ell_{out}}^{out}$ respectively, where each $\mathfrak{l}_i^{in}, \mathfrak{l}_i^{out} \in [0, \ldots, L]$ is the label associated to the given input/output wire in C^{sub}.*

   Algorithm $\mathsf{SHE.Eval}$ *associates the input ciphertexts with the input gates of C^{sub} and homomorphically evaluates C^{sub} gate by gate in an SIMD manner on the components of the input messages. For this,* $\mathsf{SHE.Eval}$ *consists of separate algorithms* $\mathsf{SHE.Add}$ *and* $\mathsf{SHE.Mult}$ *for homomorphically evaluating addition and multiplication gates respectively. More specifically, given two ciphertexts $(\mathfrak{c}_1, \mathfrak{l}_1)$ and $(\mathfrak{c}_2, \mathfrak{l}_2)$ with associated levels $\mathfrak{l}_1$ and $\mathfrak{l}_2$ respectively where $\mathfrak{l}_1, \mathfrak{l}_2 \in [0, \ldots, L]$ then*[2]*:*

   – $\mathsf{SHE.Add}_{\mathsf{ek}}((\mathfrak{c}_1, \mathfrak{l}_1), (\mathfrak{c}_2, \mathfrak{l}_2)) \to (\mathfrak{c}_{\mathsf{Add}}, \min(\mathfrak{l}_1, \mathfrak{l}_2))$*: The deterministic polynomial time addition algorithm takes as input $(\mathfrak{c}_1, \mathfrak{l}_1), (\mathfrak{c}_2, \mathfrak{l}_2)$ and outputs a ciphertext $\mathfrak{c}_{\mathsf{Add}}$ with associated level* $\min(\mathfrak{l}_1, \mathfrak{l}_2)$.

[1] In the paper, unless it is explicitly specified, we assume that some randomness has been used for encryption.

[2] Without loss of generality we assume that we can perform homomorphic operations on ciphertexts of different levels, since we can always deterministically downgrade the ciphertext level of any ciphertext to any value between zero and its current value using $\mathsf{SHE.LowerLevel}_{\mathsf{ek}}$.

- $\mathsf{SHE.Mult}_{\mathsf{ek}}((\mathfrak{c}_1, \mathfrak{l}_1), (\mathfrak{c}_2, \mathfrak{l}_2)) \rightarrow (\mathfrak{c}_{\mathsf{Mult}}, \min(\mathfrak{l}_1, \mathfrak{l}_2) - 1)$*: The deterministic polynomial time multiplication algorithm takes as input* $(\mathfrak{c}_1, \mathfrak{l}_1), (\mathfrak{c}_2, \mathfrak{l}_2)$ *and outputs a ciphertext* $\mathfrak{c}_{\mathsf{Mult}}$ *with associated level* $\min(\mathfrak{l}_1, \mathfrak{l}_2) - 1$.
- $\mathsf{SHE.ScalarMult}_{\mathsf{ek}}((\mathfrak{c}_1, \mathfrak{l}_1), \mathbf{a}) \rightarrow (\mathfrak{c}_{\mathsf{Scalar}}, \mathfrak{l}_1)$*: The deterministic polynomial time scalar multiplication algorithm takes as input* $(\mathfrak{c}_1, \mathfrak{l}_1)$ *and a plaintext* $\mathbf{a} \in \mathcal{M}$ *and outputs a ciphertext* $\mathfrak{c}_{\mathsf{Scalar}}$ *with associated level* $\mathfrak{l}_1$.

7. $\mathsf{SHE.Pack}_{\mathsf{ek}}((\mathfrak{c}_1, \mathfrak{l}_1), \ldots, (\mathfrak{c}_N, \mathfrak{l}_N)) \rightarrow (\mathfrak{c}, \min(\mathfrak{l}_1, \ldots, \mathfrak{l}_N))$*: If* $\mathfrak{c}_i$ *is a ciphertext with associated plaintext* $\chi(m_i)$*, then this procedure produces a ciphertext* $(\mathfrak{c}, \min(\mathfrak{l}_1, \ldots, \mathfrak{l}_N))$ *with associated plaintext* $\mathbf{m} = (m_1, \ldots, m_N)$.
8. $\mathsf{SHE.Unpack}_{\mathsf{ek}}(\mathfrak{c}, \mathfrak{l}) \rightarrow ((\mathfrak{c}_1, \mathfrak{l}), \ldots, (\mathfrak{c}_N, \mathfrak{l}))$*: If* $\mathfrak{c}$ *is a ciphertext with associated plaintext* $\mathbf{m} = (m_1, \ldots, m_N)$*, then this procedure produces* N *ciphertexts* $(\mathfrak{c}_1, \mathfrak{l}), \ldots, (\mathfrak{c}_N, \mathfrak{l})$ *such that* $\mathfrak{c}_i$ *has associated plaintext* $\chi(m_i)$.
9. $\mathsf{SHE.LowerLevel}_{\mathsf{ek}}((\mathfrak{c}, \mathfrak{l}), \mathfrak{l}') \rightarrow (\mathfrak{c}', \mathfrak{l}')$*: This procedure, for* $\mathfrak{l}' < \mathfrak{l}$*, produces a ciphertext* $\mathfrak{c}'$ *with the same associated plaintext as* $\mathfrak{c}$*, but at level* $\mathfrak{l}'$. □

We require the following homomorphic property to be satisfied:

- *Somewhat Homomorphic SIMD Property*: Let $C^{\mathsf{sub}} : \mathbb{F}_p^{\ell_{in}} \rightarrow \mathbb{F}_p^{\ell_{out}}$ be any subcircuit of a circuit C in the family $\mathcal{C}(L)$ with respective inputs $\mathbf{m}_1, \ldots, \mathbf{m}_{\ell_{in}} \in \mathcal{M}$, such that C^{sub} when evaluated N times in an SIMD fashion on the N components of the vectors $\mathbf{m}_1, \ldots, \mathbf{m}_{\ell_{in}}$, produces N sets of ℓ_{out} output values $\hat{\mathbf{m}}_1, \ldots, \hat{\mathbf{m}}_{\ell_{out}} \in \mathcal{M}$. Moreover, for $i \in [1, \ldots, \ell_{in}]$ let $\mathfrak{c}_i$ be a ciphertext of level $\mathfrak{l}_i^{in}$ with associated plaintext vector $\mathbf{m}_i$ and let $(\hat{\mathfrak{c}}_1, \mathfrak{l}_1^{out}), \ldots, (\hat{\mathfrak{c}}_{\ell_{out}}, \mathfrak{l}_{\ell_{out}}^{out}) = \mathsf{SHE.Eval}_{\mathsf{ek}}(C^{\mathsf{sub}}, (\mathfrak{c}_1, \mathfrak{l}_1^{in}), \ldots, (\mathfrak{c}_{\ell_{in}}, \mathfrak{l}_{\ell_{in}}^{in}))$. Then the following holds with probability one for each $i \in [1, \ldots, \ell_{out}]$:
$$\mathsf{SHE.Dec}_{\mathsf{sk}}(\hat{\mathfrak{c}}_i, \mathfrak{l}_i^{out}) = \hat{\mathbf{m}}_i.$$

We also require the following security properties:

- *Key Generation Security*: Let S and D_i be the random variables which denote the probability distribution with which the secret key sk and the ith key dk_i for the distributed decryption is selected from $\mathcal{SK}$ and $\mathcal{DK}_i$ by $\mathsf{SHE.KeyGen}$ for $i = 1, \ldots, n$. Moreover, for a set $I \subseteq \{1, \ldots, n\}$, let D_I denote the random variable which denote the probability distribution with which the set of keys for the distributed decryption, belonging to the indices in I, are selected from the corresponding $\mathcal{DK}_i$s by $\mathsf{SHE.KeyGen}$. Then the following two properties hold:
  - *Correctness*: For any set $I \subseteq \{1, \ldots, n\}$ with $|I| \geq t + 1$, $H(S|D_I) = 0$. Here $H(X|Y)$ denotes the conditional entropy of a random variable X with respect to a random variable Y [13].
  - *Privacy*: For any set $I \subset \{1, \ldots, n\}$ with $|I| \leq t$, $H(S|D_I) = H(S)$.
- *Semantic Security*: For every set $I \subset \{1, \ldots, n\}$ with $|I| \leq t$ and all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following game is negligible in κ:
  - *Key Generation*: The challenger runs $\mathsf{SHE.KeyGen}(1^{\kappa}, 1^{\mathsf{sec}}, n, t)$ to obtain $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk}, \mathsf{dk}_1, \ldots, \mathsf{dk}_n)$ and sends pk, ek and $\{\mathsf{dk}_i\}_{i \in I}$ to $\mathcal{A}$.
  - *Challenge*: $\mathcal{A}$ sends plaintexts $\mathbf{m}_0, \mathbf{m}_1 \in \mathcal{M}$ to the challenger, who randomly selects $b \in \{0, 1\}$ and sends $(\mathfrak{c}, L) = \mathsf{SHE.Enc}_{\mathsf{pk}}(\mathbf{m}_b, r)$ for some randomness r to $\mathcal{A}$.

- *Output*: $\mathcal{A}$ outputs b'.

  The advantage of $\mathcal{A}$ in the above game is defined to be $|\frac{1}{2} - \Pr[b' = b]|$.

– *Correct Share Decryption*: For any $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk}, \mathsf{dk}_1, \ldots, \mathsf{dk}_n)$ obtained as the output of $\mathsf{SHE.KeyGen}$, the following should hold for any ciphertext $(\mathfrak{c}, \mathfrak{l})$ with associated level $\mathfrak{l} \in [0, \ldots, L]$:

$$\mathsf{SHE.Dec}_{\mathsf{sk}}(\mathfrak{c}, \mathfrak{l}) = \mathsf{SHE.ShareCombine}((\mathfrak{c}, \mathfrak{l}), \{\mathsf{SHE.ShareDec}_{\mathsf{dk}_i}(\mathfrak{c}, \mathfrak{l})\}_{i \in [1, \ldots, n]}).$$

– *Share Simulation Indistinguishability*: There exists a PPT simulator $\mathsf{SHE.ShareSim}$, which on input a subset $I \subset \{1, \ldots, n\}$ of size at most t, a ciphertext $(\mathfrak{c}, \mathfrak{l})$ of level $\mathfrak{l} \in [0, \ldots, L]$, a plaintext $\mathbf{m}$ and $|I|$ decryption shares $\{\bar{\mu}_i\}_{i \in I}$ outputs $n - |I|$ simulated decryption shares $\{\bar{\mu}_j^*\}_{j \in \overline{I}}$ with the following property: For any $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk}, \mathsf{dk}_1, \ldots, \mathsf{dk}_n)$ obtained as the output of $\mathsf{SHE.KeyGen}$, any subset $I \subset \{1, \ldots, n\}$ of size at most t, any $\mathbf{m} \in \mathcal{M}$ and any $(\mathfrak{c}, \mathfrak{l})$ where $\mathbf{m} = \mathsf{SHE.Dec}_{\mathsf{sk}}(\mathfrak{c}, \mathfrak{l})$, the following distributions are statistically indistinguishable:

$$(\{\bar{\mu}_i\}_{i \in I}, \mathsf{SHE.ShareSim}((\mathfrak{c}, \mathfrak{l}), \mathbf{m}, \{\bar{\mu}_i\}_{i \in I})) \overset{s}{\approx} \left(\{\bar{\mu}_i\}_{i \in I}, \{\bar{\mu}_j\}_{j \in \overline{I}}\right),$$

where for all $i \in [1, \ldots, n]$, $\bar{\mu}_i = \mathsf{SHE.ShareDec}_{\mathsf{dk}_i}(\mathfrak{c}, \mathfrak{l})$. We require in particular that the statistical distance between the two distributions is bounded by $2^{-\mathsf{sec}}$. Moreover

$$\mathsf{SHE.ShareCombine}((\mathfrak{c}, \mathfrak{l}), \{\bar{\mu}_i\}_{i \in I} \cup \mathsf{SHE.ShareSim}((\mathfrak{c}, \mathfrak{l}), \mathbf{m}, \{\bar{\mu}_i\}_{i \in I}))$$

outputs the result $\mathbf{m}$. Here $\overline{I}$ denotes the complement of the set I; i.e. $\overline{I} = \{1, \ldots, n\} \setminus I$.

In the full version of the paper we instantiate the abstract syntax with a threshold SHE scheme based on the BGV scheme [7]. We pause to note the difference between our underlying SHE, which is just an SHE scheme which supports distributed decryption, and that of [1] which requires a special key homomorphic FHE scheme.

4 MPC from SHE – The Semi-honest Settings

In this section we present our generic MPC protocol for the computation of any arbitrary depth d circuit using an abstract threshold L-levelled SHE scheme. For the ease of exposition we first concentrate on the case of semi-honest security, and then we deal with active security in Section 5.

Without loss of generality we make the simplifying assumption that the function f to be computed takes a single input from each party and has a single output; specifically $f : \mathbb{F}_p^n \rightarrow \mathbb{F}_p$. The ideal functionality $\mathcal{F}_f$ presented in Figure 1 computes such a given function f, represented by a well formed circuit C. We will present a protocol to realise the ideal functionality $\mathcal{F}_f$ in a hybrid model in which we are given access to an ideal functionality $\mathcal{F}_{\text{SETUPGEN}}$ which implements a distributed key generation for the underlying SHE scheme. In particular the $\mathcal{F}_{\text{SETUPGEN}}$ functionality presented in Figure 2 computes the public key, secret key, evaluation key and the keys for the distributed decryption of an L-levelled SHE scheme, distributes the public key and the evaluation key

Functionality $\mathcal{F}_f$

$\mathcal{F}_f$ interacts with the parties $P_1, \ldots, P_n$ and the adversary $\mathcal{S}$ and is parametrized by an n-input function $f : \mathbb{F}_p^n \to \mathbb{F}_p$.

– Upon receiving (sid, i, x_i) from the party P_i for every $i \in [1, \ldots, n]$ where $x_i \in \mathbb{F}_p$, compute $y = C(x_1, \ldots, x_n)$, send (sid, y) to all the parties and the adversary $\mathcal{S}$ and halt. Here C denotes the (publicly known) well formed arithmetic circuit over $\mathbb{F}_p$ representing the function f.

Fig. 1. The Ideal Functionality for Computing a Given Function

to all the parties and sends the ith key dk_i (for the distributed decryption) to the party P_i for each $i \in [1, \ldots, n]$. In addition, the functionality also computes a random encryption $\mathfrak{c}_{\underline{\mathbf{1}}}$ with associated plaintext $\underline{\mathbf{1}} = (1, \ldots, 1) \in \mathcal{M}$ and sends it to all the parties. Looking ahead, $\mathfrak{c}_{\underline{\mathbf{1}}}$ will be required while proving the security of our MPC protocol. The ciphertext $\mathfrak{c}_{\underline{\mathbf{1}}}$ is at level one, as we only need it to pre-multiply the ciphertexts which are going to be decrypted via the distributed decryption protocol; thus the output of a multiplication by $\mathfrak{c}_{\underline{\mathbf{1}}}$ need only be at level zero. Looking ahead, this ensures that (with respect to our instantiation of SHE) the noise is kept to a minimum at this stage of the protocol.

Functionality $\mathcal{F}_{\text{SETUPGEN}}$

$\mathcal{F}_{\text{SETUPGEN}}$ interacts with the parties $P_1, \ldots, P_n$ and the adversary $\mathcal{S}$ and is parametrized by an L-levelled SHE scheme.

– Upon receiving (sid, i) from the party P_i for every $i \in [1, \ldots, n]$, compute $(\mathsf{pk}, \mathsf{ek}, \mathsf{sk}, \mathsf{dk}_1, \ldots, \mathsf{dk}_n) = \mathsf{SHE.KeyGen}(1^\kappa, 1^{\mathsf{sec}}, n, t)$ and $(\mathfrak{c}_{\underline{\mathbf{1}}}, 1) = \mathsf{SHE.LowerLevel}_{\mathsf{ek}}((\mathsf{SHE.Enc}_{\mathsf{pk}}(\underline{\mathbf{1}}, r), 1)$ for $\underline{\mathbf{1}} = (1, \ldots, 1) \in \mathcal{M}$ and some randomness r. Finally send $(\mathsf{sid}, \mathsf{pk}, \mathsf{ek}, \mathsf{dk}_i, (\mathfrak{c}_{\underline{\mathbf{1}}}, 1))$ to the party P_i for every $i \in [1, \ldots, n]$ and halt.

Fig. 2. The Ideal Functionality for Key Generation

4.1 The MPC Protocol in the $\mathcal{F}_{\text{SETUPGEN}}$-hybrid Model

Here we present our MPC protocol Π_f^{SH} in the $\mathcal{F}_{\text{SETUPGEN}}$-hybrid model. Let C be the (well formed) arithmetic circuit representing the function f and C^{aug} be the associated augmented circuit (which includes the necessary $\mathsf{Refresh}$ gates). The protocol Π_f^{SH} (see Figure 3) runs in two phases: offline and online. The computation performed in the offline phase is completely independent of the circuit and (private) inputs of the parties and therefore can be carried out well ahead of the time (namely the online phase) when the function and inputs are known. If the parties have more than one input/output then one can apply packing/unpacking at the input/output stages of the protocol; we leave this minor modification to the reader.

In the offline phase, the parties interact with $\mathcal{F}_{\text{SETUPGEN}}$ to obtain the public key, evaluation key and their respective keys for performing distributed decryption, corresponding to a threshold L-levelled SHE scheme. Next each party sends encryptions of ζ random elements and then additively combines them (by applying the homomorphic addition to the ciphertexts encrypting the random elements) to generate ζ ciphertexts at level L of truly random elements (unknown to the adversary). Here ζ is assumed to be large enough, so that for a typical circuit it is more than the number of refresh gates in the circuit, i.e. $\zeta > \mathbb{G}_R$. Looking ahead, these random ciphertexts created in the offline phase are used in the online phase to evaluate refresh gates by (homomorphically) masking the messages associated with the input wires of a refresh gate.

During the online phase, the parties encrypt their private inputs and distribute the corresponding ciphertexts to all other parties. These ciphertexts are transmitted at level one, thus consuming low bandwidth, and are then elevated to level L by the use of a following Refresh gate (which would have been inserted by the circuit augmentation process). Note that the inputs of the parties are in $\mathbb{F}_p$ and so the parties first apply the mapping χ (embedding $\mathbb{F}_p$ into the message space $\mathcal{M}$ of SHE) before encrypting their private inputs.

The input stage is followed by the homomorphic evaluation of C^{aug} as follows: The addition and multiplication gates are evaluated locally using the addition and multiplication algorithm of the SHE. For each refresh gate, the parties execute the following protocol to enable a "multiparty bootstrapping" of the input ciphertexts: the parties pick one of the random ciphertext created in the offline phase (for each refresh gate a different ciphertext is used) and perform the following computation to refresh N ciphertexts with levels in the range $[1, \ldots, L]$ and obtain N fresh level L ciphertexts, with the associated messages unperturbed:

- Let $(\mathfrak{c}_1, \mathfrak{l}_1), \ldots, (\mathfrak{c}_N, \mathfrak{l}_N)$ be the N ciphertexts with associated plaintexts $\chi(z_1), \ldots, \chi(z_N)$ with every $z_i \in \mathbb{F}_p$, that need to be refreshed (i.e. they are the inputs of a refresh gate).
- The N ciphertexts are then (locally) packed into a single ciphertext $\mathfrak{c}$, which is then homomorphically masked with a random ciphertext from the offline phase.
- The resulting masked ciphertext is then publicly opened via distributed decryption. This allows for the creation of a fresh encryption of the opened value at level L.
- The resulting fresh encryption is then homomorphically unmasked so that its associated plaintext is the same as original plaintext prior to the original masking.
- This fresh (unmasked) ciphertext is then unpacked to obtain N fresh ciphertexts, having the same associated plaintexts as the original N ciphertexts $\mathfrak{c}_i$ but at level L.

By packing the ciphertexts together we only need to invoke distributed decryption once, instead of N times. This leads to a more communication efficient online phase, since the distributed decryption is the only operation that demands communication. Without

Protocol Π_f^{SH}

Let C^{aug} denote an augmented circuit for a well formed circuit C over $\mathbb{F}_p$ representing f and let SHE be a threshold L-levelled SHE. Moreover, let $\mathcal{P} = \{P_1, \dots, P_n\}$ be the set of n parties. For the session ID sid the parties do the following:

Offline Computation: Every party $P_i \in \mathcal{P}$ does the following:

- Call $\mathcal{F}_{\text{SETUPGEN}}$ with (sid, i) and receive $(\mathsf{sid}, \mathsf{pk}, \mathsf{ek}, \mathsf{dk}_i, (\mathfrak{c}_{\underline{1}}, 1))$.
- Randomly select ζ plaintexts $\mathbf{m}_{i,1}, \dots, \mathbf{m}_{i,\zeta} \in \mathcal{M}$, and compute $(\mathfrak{c}_{\mathbf{m}_{i,k}}, L) = \mathsf{SHE.Enc}_{\mathsf{pk}}(\mathbf{m}_{i,k}, r_{i,k})$. Send $(\mathsf{sid}, i, (\mathfrak{c}_{\mathbf{m}_{i,1}}, L), \dots, (\mathfrak{c}_{\mathbf{m}_{i,\zeta}}, L))$ to all parties in $\mathcal{P}$.
- Upon receiving $(\mathsf{sid}, j, (\mathfrak{c}_{\mathbf{m}_{j,1}}, L), \dots, (\mathfrak{c}_{\mathbf{m}_{j,\zeta}}, L))$ from all parties $P_j \in \mathcal{P}$, apply $\mathsf{SHE.Add}$ for $1 \le k \le \zeta$, on $(\mathfrak{c}_{\mathbf{m}_{1,k}}, L), \dots, (\mathfrak{c}_{\mathbf{m}_{n,k}}, L)$, set the resultant ciphertext as the kth offline ciphertext $\mathfrak{c}_{\mathbf{m}_k}$ with the (unknown) associated plaintext $\mathbf{m}_k = \mathbf{m}_{1,k} + \cdots + \mathbf{m}_{n,k}$.

Online Computation: Every party $P_i \in \mathcal{P}$ does the following:

- **Input Stage**: On having input $x_i \in \mathbb{F}_p$, compute $(\mathfrak{c}_{\mathbf{x}_i}, 1) = \mathsf{SHE.LowerLevel}_{\mathsf{ek}}(\mathsf{SHE.Enc}_{\mathsf{pk}}(\chi(x_i), r_i), 1)$ with randomness r_i and send $(\mathsf{sid}, i, (\mathfrak{c}_{\mathbf{x}_i}, 1))$ to each party. Receive $(\mathsf{sid}, j, (\mathfrak{c}_{\mathbf{x}_j}, 1))$ from each party $P_j \in \mathcal{P}$.
- **Computation Stage**: Associate the received ciphertexts with the corresponding input wires of C^{aug} and then homomorphically evaluate the circuit C^{aug} gate by gate as follows:
  - **Addition Gate and Multiplication Gate**: Given $(\mathfrak{c}_1, \mathfrak{l}_1)$ and $(\mathfrak{c}_2, \mathfrak{l}_2)$ associated with the input wires of the gate where $\mathfrak{l}_1, \mathfrak{l}_2 \in [1, \dots, L]$, locally compute $(\mathfrak{c}, \mathfrak{l}) = \mathsf{SHE.Add}_{\mathsf{ek}}((\mathfrak{c}_1, \mathfrak{l}_1), (\mathfrak{c}_2, \mathfrak{l}_2))$ with $\mathfrak{l} = \min(\mathfrak{l}_1, \mathfrak{l}_2)$ for an addition gate and $(\mathfrak{c}, \mathfrak{l}) = \mathsf{SHE.Mult}_{\mathsf{ek}}((\mathfrak{c}_1, \mathfrak{l}_1), (\mathfrak{c}_2, \mathfrak{l}_2))$ with $\mathfrak{l} = \min(\mathfrak{l}_1, \mathfrak{l}_2) - 1$ for a multiplication gate; for the multiplication gate, $\mathfrak{l}_1, \mathfrak{l}_2 \in [2, \dots, L]$, instead of $[1, \dots, L]$. Associate $(\mathfrak{c}, \mathfrak{l})$ with the output wire of the gate.
  - **Refresh Gate**: For the kth refresh gate in the circuit, the kth offline ciphertext $(\mathfrak{c}_{\mathbf{m}_k}, L)$ is used. Let $(\mathfrak{c}_1, \mathfrak{l}_1), \dots, (\mathfrak{c}_N, \mathfrak{l}_N)$ be the ciphertexts associated with the input wires of the refresh gate where $\mathfrak{l}_1, \dots, \mathfrak{l}_N \in [1, \dots, L]$:
    - **Packing**: Locally compute $(\mathfrak{c}_{\mathbf{z}}, \mathfrak{l}) = \mathsf{SHE.Pack}_{\mathsf{ek}}(\{(\mathfrak{c}_i, \mathfrak{l}_i)\}_{i \in [1,\dots,N]})$ where $\mathfrak{l} = \min(\mathfrak{l}_1, \dots, \mathfrak{l}_N)$.
    - **Masking**: Locally compute $(\mathfrak{c}_{\mathbf{z}+\mathbf{m}_k}, 0) = \mathsf{SHE.Add}_{\mathsf{ek}}(\mathsf{SHE.Mult}_{\mathsf{ek}}((\mathfrak{c}_{\mathbf{z}}, \mathfrak{l}), (\mathfrak{c}_{\underline{1}}, 1)), (\mathfrak{c}_{\mathbf{m}_k}, L))$
    - **Decrypting**: Locally compute the decryption share $\bar{\mu}_i = \mathsf{SHE.ShareDec}_{\mathsf{dk}_i}(\mathfrak{c}_{\mathbf{z}+\mathbf{m}_k}, 0)$ and send $(\mathsf{sid}, i, \bar{\mu}_i)$ to every other party. On receiving $(\mathsf{sid}, j, \bar{\mu}_j)$ from every $P_j \in \mathcal{P}$, compute the plaintext $\mathbf{z} + \mathbf{m}_k = \mathsf{SHE.ShareCombine}((\mathfrak{c}_{\mathbf{z}+\mathbf{m}_k}, 0), \{\bar{\mu}_j\}_{j \in [1,\dots,n]})$.
    - **Re-encrypting**: Locally re-encrypt $\mathbf{z} + \mathbf{m}_k$ by computing $(\hat{\mathfrak{c}}_{\mathbf{z}+\mathbf{m}_k}, L) = \mathsf{SHE.Enc}_{\mathsf{pk}}(\mathbf{z} + \mathbf{m}_k, r)$ using a publicly known (common) randomness r, (This can simply be the zero string for our BGV instantiation, we only need to map the known plaintext into a ciphertext element).
    - **Unmasking**: Locally subtract $(\mathfrak{c}_{\mathbf{m}_k}, L)$ from $(\hat{\mathfrak{c}}_{\mathbf{z}+\mathbf{m}_k}, L)$ to obtain $(\hat{\mathfrak{c}}_{\mathbf{z}}, L)$.
    - **Unpacking**: Locally compute $(\hat{\mathfrak{c}}_1, L), \dots, (\hat{\mathfrak{c}}_N, L) = \mathsf{SHE.Unpack}_{\mathsf{ek}}(\hat{\mathfrak{c}}_{\mathbf{z}}, L)$ and associate $(\hat{\mathfrak{c}}_1, L), \dots, (\hat{\mathfrak{c}}_N, L)$ with the output wires of the refresh gate.
- **Output Stage**: Let $(\mathfrak{c}, \mathfrak{l})$ be the ciphertext associated with the output wire of C^{aug} where $\mathfrak{l} \in [1, \dots, L]$.
  - **Randomization:** Compute a random encryption $(\mathfrak{c}_i, L) = \mathsf{SHE.Enc}_{\mathsf{pk}}(\underline{\mathbf{0}}, r_i')$ of $\underline{\mathbf{0}} = (0, \dots, 0)$ and send $(\mathsf{sid}, i, (\mathfrak{c}_i, L))$ to every other party. On receiving $(\mathsf{sid}, j, (\mathfrak{c}_j, L))$ from every $P_j \in \mathcal{P}$, apply $\mathsf{SHE.Add}$ on $\{(\mathfrak{c}_j, L)\}_{j \in [1,\dots,n]}$ to obtain $(\mathfrak{c}_{\underline{\mathbf{0}}}, L)$. Compute $(\hat{\mathfrak{c}}, 0) = \mathsf{SHE.Add}_{\mathsf{ek}}(\mathsf{SHE.Mult}_{\mathsf{ek}}((\mathfrak{c}, \mathfrak{l}), (\mathfrak{c}_{\underline{1}}, 1)), (\mathfrak{c}_{\underline{\mathbf{0}}}, L))$.
  - **Output Decryption:** Compute $\bar{\gamma}_i = \mathsf{SHE.ShareDec}_{\mathsf{dk}_i}(\hat{\mathfrak{c}}, 0)$ and send $(\mathsf{sid}, i, \bar{\gamma}_i)$ to every party. On receiving $(\mathsf{sid}, j, \bar{\gamma}_j)$ from every $P_j \in \mathcal{P}$, compute $\mathbf{y} = \mathsf{SHE.ShareCombine}((\hat{\mathfrak{c}}, 0), \{\bar{\gamma}_j\}_{j \in [1,\dots,n]})$, output y and halt, where $y = \chi^{-1}(\mathbf{y})$.

Fig. 3. The Protocol for Realizing $\mathcal{F}_f$ against a Semi-Honest Adversary in the $\mathcal{F}_{\text{SETUPGEN}}$-hybrid Model

affecting the correctness of the above technique, but to ensure security, we add an additional step while doing the masking: the parties homomorphically pre-multiply the ciphertext $\mathfrak{c}$ with $\mathfrak{c}_{\underline{\mathbf{1}}}$ before masking. Recall that $\mathfrak{c}_{\underline{\mathbf{1}}}$ is an encryption of $\underline{\mathbf{1}} \in \mathcal{M}$ generated by $\mathcal{F}_{\text{SETUPGEN}}$ and so by doing the above operation, the plaintext associated with $\mathfrak{c}$ remains the same. During the simulation in the security proof, this step allows the simulator to set the decrypted value to the random mask (irrespective of the circuit inputs), by playing the role of $\mathcal{F}_{\text{SETUPGEN}}$ and replacing $\mathfrak{c}_{\underline{\mathbf{1}}}$ with $\mathfrak{c}_{\underline{\mathbf{0}}}$, a random encryption of $\underline{\mathbf{0}} = (0, \ldots, 0)$. Furthermore, this step explains the reason why we made provision for an extra multiplication during circuit augmentation by insisting that the refresh gates take inputs with labels in $[1, \ldots, L]$, instead of $[0, \ldots, L]$; the details are available in the simulation proof of security of our MPC protocol.

Finally, the function output y is obtained by another distributed decryption of the output ciphertext. However, this step is also not secure unless the ciphertext is randomized again by pre-multiplication by $\mathfrak{c}_{\underline{\mathbf{1}}}$ and adding n encryptions of $\underline{\mathbf{0}}$ where each party contributes one encryption. In the simulation, the simulator gives encryption of $\chi(y)$ on behalf of one honest party and replaces $\mathfrak{c}_{\underline{\mathbf{1}}}$ by $\mathfrak{c}_{\underline{\mathbf{0}}}$, letting the output ciphertext correspond to the actual output y, even though the circuit is evaluated with zero as the inputs of the honest parties during the simulation (the simulator will not know the real inputs of the honest parties and thus will simulate them with zero). A similar idea was also used in [19]; details can be found in the security proof.

Intuitively, privacy follows because at any stage of the computation, the keys of the honest parties for the distributed decryption are not revealed and so the adversary will not be able to decrypt any intermediate ciphertext. Correctness follows from the properties of the SHE and the fact that the level of each ciphertext in the protocol remains in the range $[1, \ldots, L]$, thanks to the refresh gates. So even though the circuit C may have any arbitrary depth $d > L$, we can homomorphically evaluate C using an L-levelled SHE.

Theorem 1. *Let $f : \mathbb{F}_p^n \rightarrow \mathbb{F}_p$ be a function over $\mathbb{F}_p$ represented by a well formed arithmetic circuit C of depth d over $\mathbb{F}_p$. Let $\mathcal{F}_f$ (presented in Figure 1) be the ideal functionality computing f and let* SHE *be a threshold L-levelled SHE scheme. Then the protocol Π_f^{SH} UC-secure realizes $\mathcal{F}_f$ against a static, semi-honest adversary $\mathcal{A}$, corrupting upto $t < n$ parties in the $\mathcal{F}_{\text{SETUPGEN}}$-hybrid Model.*

The proof is given in the full version of the paper.

5 MPC from SHE – The Active Setting

The functionalities from Section 4 are in the passive corruption model. In the presence of an active adversary, the functionalities will be modified as follows: the respective functionality considers the input received from the majority of the parties and performs the task it is supposed to do on those inputs. For example, in the case of $\mathcal{F}_f$, the functionality considers for the computation those x_is, corresponding to the P_is from which the functionality has received the message (sid, i, x_i); on the behalf of the remaining P_is, the functionality substitutes 0 as the default input for the computation. Similarly for $\mathcal{F}_{\text{SETUPGEN}}$, the functionality performs its task if it receives the message (sid, i) from the majority of the parties. These are the standard notions of defining ideal functionalities for various corruption scenarios and we refer [26] for the complete formal details; we will not present separately the ideal functionality $\mathcal{F}_f$ and $\mathcal{F}_{\text{SETUPGEN}}$ for the malicious setting.

A closer look at Π_f^{SH} shows that we can "compile" it into an actively secure MPC protocol tolerating t active corruptions if we ensure that every corrupted party "proves" in a zero knowledge (ZK) fashion that it constructed the following correctly: **(1)** The ciphertexts in the offline phase; **(2)** The ciphertexts during the input stage and **(3)** The randomizing ciphertexts during the output stage.

Apart from the above three requirements, we also require a "robust" version of the SHE.ShareCombine method which works correctly even if up to t input decryption shares are incorrect. In the full version we show that for our specific SHE scheme, the SHE.ShareCombine algorithm (based on the standard error-correction) is indeed robust, provided $t < n/3$. For the case of $t < n/2$ we also show that by including additional steps and zero-knowledge proofs (namely proof of correct decryption), one can also obtain a robust output. Interestingly the MPC protocol requires the transmission of at most $\mathcal{O}(n^3)$ such additional zero-knowledge proofs; i.e. the communication needed to obtain robustness is *independent* of the circuit. We stress that $t < n/2$ is the *optimal resilience* for computationally secure MPC against active corruptions (with robustness and fairness) [12, 27]. To keep the protocol presentation and its proof simple, we assume a robust SHE.ShareCombine (i.e. for the case of $t < n/3$), which applies error correction for the correct decryption.

Functionality $\mathcal{F}_{\text{ZK}}^{R}$

$\mathcal{F}_{\text{ZK}}^{R}$ interacts with a prover $P_i \in \{P_1, \ldots, P_n\}$ and the set of n verifiers $\mathcal{P} = \{P_1, \ldots, P_n\}$ and the adversary $\mathcal{S}$.

– Upon receiving $(\mathsf{sid}, i, (x, w))$ from the prover $P_i \in \{P_1, \ldots, P_n\}$, the functionality sends (sid, i, x) to all the verifiers in $\mathcal{P}$ and $\mathcal{S}$ if $R(x, w)$ is true. Else it sends $(\mathsf{sid}, i, \perp)$ and halts.

Fig. 4. The Ideal Functionality for ZK

The actively secure MPC protocol is given in Figure 4, it uses an ideal ZK functionality $\mathcal{F}_{\text{ZK}}^{R}$, parametrized with an NP-relation R. We apply this ZK functionality to the following relations to obtain the functionalities $\mathcal{F}_{\text{ZK}}^{R_{enc}}$ and $\mathcal{F}_{\text{ZK}}^{R_{zeroenc}}$. We note that UC-secure realizations of $\mathcal{F}_{\text{ZK}}^{R_{enc}}$ and $\mathcal{F}_{\text{ZK}}^{R_{zeroenc}}$ can be obtained in the CRS model, similar techniques to these are used in [2]. Finally we do not worry about the instantiation of $\mathcal{F}_{\text{SETUPGEN}}$ as we consider it a one time set-up, which can be done via standard techniques (such as running an MPC protocol).

– $R_{enc} = \{((\mathfrak{c}, \mathfrak{l}), (\mathbf{x}, r)) \mid (\mathfrak{c}, \mathfrak{l}) = \mathsf{SHE.Enc}_{\mathsf{pk}}(\mathbf{x}, r)$ if $\mathfrak{l} = L \quad \vee \quad (\mathfrak{c}, \mathfrak{l}) = \mathsf{SHE.LowerLevel}_{\mathsf{ek}}(\mathsf{SHE.Enc}_{\mathsf{pk}}(\mathbf{x}, r), 1)$ if $\mathfrak{l} = 1\}$: we require this relation to hold

Protocol $\mathit{\Pi}_f^{\text{MAL}}$

Let C be the well formed arithmetic circuit over $\mathbb{F}_p$ representing the function f, let C^{aug} denote an augmented circuit associated with C, and let SHE be a threshold L-levelled SHE scheme. For session ID sid the parties in $\mathcal{P} = \{P_1, \ldots, P_n\}$ do the following:

Offline Computation: Every party $P_i \in \mathcal{P}$ does the following:

- Call $\mathcal{F}_{\text{SETUPGEN}}$ with (sid, i) and receive $(\mathsf{sid}, \mathsf{pk}, \mathsf{ek}, \mathsf{dk}_i, (\mathfrak{c}_{\underline{\mathbf{1}}}, 1))$.
- Do the same as in the offline phase of $\mathit{\Pi}_f^{\text{SH}}$, except that for every message $\mathbf{m}_{ik}$ for $k \in [1, \ldots, \zeta]$ and the corresponding ciphertext $(\mathfrak{c}_{\mathbf{m}_{ik}}, L) = \mathsf{SHE.Enc}_{\mathsf{pk}}(\mathbf{m}_{ik}, r_{ik})$, call $\mathcal{F}_{\text{ZK}}^{R_{enc}}$ with $(\mathsf{sid}, i, ((\mathfrak{c}_{\mathbf{m}_{ik}}, L), (\mathbf{m}_{ik}, r_{ik})))$. Receive $(\mathsf{sid}, j, (\mathfrak{c}_{\mathbf{m}_{jk}}, L))$ from $\mathcal{F}_{\text{ZK}}^{R_{enc}}$ for $k \in [1, \ldots, \zeta]$ corresponding to each $P_j \in \mathcal{P}$. If $(\mathsf{sid}, j, \perp)$ is received from $\mathcal{F}_{\text{ZK}}^{R_{enc}}$ for some $P_j \in \mathcal{P}$, then consider ζ publicly known level L encryptions of random values from $\mathcal{M}$ as $(\mathfrak{c}_{\mathbf{m}_{jk}}, L)$ for $k \in [1, \ldots, \zeta]$.

Online Computation: Every party $P_i \in \mathcal{P}$ does the following:

- **Input Stage**: On having input $x_i \in \mathbb{F}_p$, compute level L ciphertext $(\mathfrak{c}_{\mathbf{x}_i}, 1) = \mathsf{SHE.LowerLevel}_{\mathsf{ek}}(\mathsf{SHE.Enc}_{\mathsf{pk}}(\chi(x_i), r_i), 1)$ with randomness r_i and call $\mathcal{F}_{\text{ZK}}^{R_{enc}}$ with the message $(\mathsf{sid}, i, ((\mathfrak{c}_{\mathbf{x}_i}, 1), (\chi(x_i), r_i)))$. Receive $(\mathsf{sid}, j, (\mathfrak{c}_{\mathbf{x}_j}, 1))$ from $\mathcal{F}_{\text{ZK}}^{R_{enc}}$ corresponding to each $P_j \in \mathcal{P}$. If $(\mathsf{sid}, j, \perp)$ is received from $\mathcal{F}_{\text{ZK}}^{R_{enc}}$ for some $P_j \in \mathcal{P}$, then consider a publicly known level 1 encryption of $\chi(0)$ as $(\mathfrak{c}_{\mathbf{x}_j}, 1)$ for such a P_j.
- **Computation Stage**: Same as $\mathit{\Pi}_f^{\text{SH}}$, except that now the robust $\mathsf{SHE.ShareCombine}$ is used.
- **Output Stage**: Let $(\mathfrak{c}, \mathfrak{l})$ be the ciphertext associated with the output wire of C^{aug} where $\mathfrak{l} \in [1, \ldots, L]$.
  - **Randomization:** Compute a random encryption $(c_i, L) = \mathsf{SHE.Enc}_{\mathsf{pk}}(\underline{\mathbf{0}}, r'_i)$ of $\underline{\mathbf{0}} = (0, \ldots, 0)$ and call $\mathcal{F}_{\text{ZK}}^{R_{zeroenc}}$ with the message $(\mathsf{sid}, i, ((c_i, L), (\underline{\mathbf{0}}, r'_i)))$. Receive $(\mathsf{sid}, j, (c_j, L))$ from $\mathcal{F}_{\text{ZK}}^{R_{zeroenc}}$ corresponding to each $P_j \in \mathcal{P}$. If $(\mathsf{sid}, j, \perp)$ is received from $\mathcal{F}_{\text{ZK}}^{R_{zeroenc}}$ for some $P_j \in \mathcal{P}$, then consider a publicly known level L encryption of $\underline{\mathbf{0}}$ as (c_j, L) for such a P_j.
  - The rest of the steps are same as in $\mathit{\Pi}_f^{\text{SH}}$, except that now the robust $\mathsf{SHE.ShareCombine}$ is used.

Fig. 5. The Protocol for Realizing $\mathcal{F}_f$ against an Active Adversary in the $(\mathcal{F}_{\text{SETUPGEN}}, \mathcal{F}_{\text{ZK}}^{R_{enc}}, \mathcal{F}_{\text{ZK}}^{R_{zeroenc}})$-hybrid Model

for the offline stage ciphertexts (where $\mathfrak{l} = L$) and for the input stage ciphertexts (where $\mathfrak{l} = 1$).

- $R_{zeroenc} = \{((\mathfrak{c}, L), (\mathbf{x}, r)) \quad | \quad (\mathfrak{c}, L) = \mathsf{SHE.Enc}_{\mathsf{pk}}(\mathbf{x}, r) \wedge \mathbf{x} = \underline{\mathbf{0}}\}$: we require this relation to hold for the randomizing ciphertexts during the output stage.

We are now ready to present the protocol Π_f^{MAL} (see Figure 5) in the ($\mathcal{F}_{\text{SETUPGEN}}$, $\mathcal{F}_{\text{ZK}}^{R_{enc}}, \mathcal{F}_{\text{ZK}}^{R_{zeroenc}}$)-hybrid model and assuming a robust SHE.ShareCombine based on error-correction (i.e. for the case $t < n/3$).

Theorem 2. *Let $f : \mathbb{F}_p^n \to \mathbb{F}_p$ be a function represented by a well-formed arithmetic circuit C over $\mathbb{F}_p$. Let $\mathcal{F}_f$ (presented in Figure 1) be the ideal functionality computing f and let* SHE *be a threshold L-levelled SHE scheme such that* SHE.ShareCombine *is robust. Then the protocol Π_f^{MAL} UC-secure realises $\mathcal{F}_f$ in the $(\mathcal{F}_{\text{SETUPGEN}}, \mathcal{F}_{\text{ZK}}^{R_{enc}}$, $\mathcal{F}_{\text{ZK}}^{R_{zeroenc}})$-hybrid Model against a static, active adversary $\mathcal{A}$ corrupting t parties.*

See the full version for a proof of this theorem.

6 Estimating the Consumed Bandwidth

In the full version we determine the parameters for the instantiation of our SHE scheme using BGV by adapting the analysis from [18, 25]. In this section we use this parameter estimation to show that our MPC protocol can in fact give improved communication complexity compared to the standard MPC protocols, for relatively small values of the parameter L. We are interested in the communication cost of our online stage computation. To ease our exposition we will focus on the passively secure case from Section 4; the analysis for the active security case with $t < n/3$ is exactly the same (bar the additional cost of the exchange of zero-knowledge proofs for the input stage and the output stage). For the case of active security with $t < n/2$ we also need to add in the communication related to the dispute control strategy outlined in the full version for attaining robust SHE.ShareCombine with $t < n/2$; but this is a cost which is proportional to $\mathcal{O}(n^3)$.

To get a feel for the parameters we now specialise the BGV instantiation from the full version of this paper to the case of finite fields of size $p \approx 2^{64}$, statistical security parameter sec of 40, and for various values of the computational security level κ. We estimate in Table 1 the value of N, assuming a small value for n (we need to restrict to small n to ensure a large enough range in the PRF needed in the distributed decryption protocol; see the full version).

Since a Refresh gate requires the transmission of $n-1$ elements (namely the decryption shares) in the ring R_{q_0} from party P_i to the other parties, the total communication in our protocol (in bits) is

$$|\mathbb{G}_R| \cdot n \cdot (n-1) \cdot |R_{q_0}|,$$

where $|R_{q_0}|$ is the number of bits needed to transmit an element in R_{q_0}, i.e. $N \cdot \log_2 p_0$. Assuming the circuit meets our requirement of being well formed, this implies that total communication cost for our protocol is

$$\frac{2 \cdot |\mathbb{G}_M| \cdot n \cdot (n-1) \cdot N \cdot \log_2 p_0}{L \cdot N} = \frac{2 \cdot n \cdot (n-1) \cdot |\mathbb{G}_M|}{L} \cdot \log_2(309 \cdot 2^{\text{sec}} \cdot p \cdot \sqrt{N}).$$

L	$\kappa = 80$	$\kappa = 128$	$\kappa = 256$
2	16384	16384	32768
3	16384	16384	32768
4	16384	32768	32768
5	32768	32768	65536
6	32768	32768	65536
7	32768	32768	65536
8	32768	65536	65536
9	32768	65536	65536
10	65536	65536	65536

Table 1. The value of N for various values of κ and L

Using the batch distributed decryption technique (of efficiently and parallely evaluating $t + 1$ independent Refresh gates simultaneously) from the full version this can be reduced to

$$\mathsf{Cost} = \frac{4 \cdot n \cdot (n-1) \cdot |\mathbb{G}_M|}{L \cdot (t+1)} \cdot \log_2(309 \cdot 2^{\text{sec}} \cdot p \cdot \sqrt{N}).$$

We are interested in the *overhead per multiplication gate*, in terms of equivalent numbers of finite field elements in $\mathbb{F}_p$, which is given by $\mathsf{Cost}/(|\mathbb{G}_M| \cdot \log_2 p)$, and the cost per party is $\mathsf{Cost}/(|\mathbb{G}_M| \cdot n \cdot \log_2 p)$.

At the 128 bit security level, with $p \approx 2^{64}$, and sec $= 40$ (along with the above estimated values of N), this means for $n = 3$ parties, and at most $t = 1$ corruption, we obtain the following cost estimates:

L		2	3	4	5	6	7	8	9	10
Total Cost	$\mathsf{Cost}/(|\mathbb{G}_M| \cdot \log_2 p)$	12.49	8.33	6.31	5.05	4.21	3.61	3.19	2.84	2.55
Per party Cost	$\mathsf{Cost}/(|\mathbb{G}_M| \cdot n \cdot \log_2 p)$	4.16	2.77	2.10	1.68	1.40	1.20	1.06	0.94	0.85

Note for $L = 2$ our protocol becomes the one which requires interaction after every multiplication, for $L = 3$ we require interaction only after every two multiplications and so on. Note that most practical MPC protocols in the preprocessing model have a per gate per party communication cost of at least 2 finite field elements, e.g. [20]. Thus, even when $L = 5$, we obtain better communication efficiency in the online phase than traditional practical protocols in the preprocessing model with these parameters.

7 Acknowledgements

This work has been supported in part by ERC Advanced Grant ERC-2010-AdG-267188-CRIPTO, by EPSRC via grant EP/I03126X, and by Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under agreement number FA8750-11-2-0079[3]. The second author was supported by an Trend Micro Ltd, and the fifth author was supported by in part by a Royal Society Wolfson Merit Award.

References

1. G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 483–501, 2012.
2. G. Asharov, A. Jain, and D. Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. *IACR Cryptology ePrint Archive*, 2011:613, 2011.
3. E. Ben-Sasson, S. Fehr, and R. Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 663–680, 2012.
4. R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias. Semi-homomorphic encryption and multiparty computation. In *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 169–188, 2011.
5. D. Bogdanov, S. Laur, and J. Willemson. Sharemind: A framework for fast privacy-preserving computations. In *ESORICS*, volume 5283 of *Lecture Notes in Computer Science*, pages 192–206, 2008.
6. Z. Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886, 2012.
7. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325. ACM, 2012.
8. Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106. IEEE, 2011.
9. Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524, 2011.
10. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.
11. O. Catrina and A. Saxena. Secure computation with fixed-point numbers. In *Financial Cryptography*, volume 6052 of *Lecture Notes in Computer Science*, pages 35–50, 2010.
12. R. Cleve. Limits on the security of coin flips when half the processors are faulty (Extended abstract). In *STOC*, pages 364–369. ACM, 1986.
13. T. M. Cover and J. A. Thomas. *Elements of Information theory*. Wiley, 2006.
14. I. Damgård, M. Fitzi, E. Kiltz, J.B. Nielsen, and T. Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 285–304, 2006.
15. I. Damgård, Y. Ishai, M. Krøigaard, J.B. Nielsen, and A. Smith. Scalable multiparty computation with nearly optimal work and resilience. In *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 241–261, 2008.

[3] The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Defense Advanced Research Projects Agency (DARPA) or the U.S. Government.

16. I. Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 445–465, 2010.
17. I. Damgård, M. Keller, E. Larraia, C. Miles, and N.P. Smart. Implementing AES via an actively/covertly secure dishonest-majority mpc protocol. In *SCN*, volume 7485 of *Lecture Notes in Computer Science*, pages 241–263, 2012.
18. I. Damgard, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart. Practical covertly secure MPC for dishonest majority – or: Breaking the SPDZ limits. In *ESORICS*, volume 8134 of *Lecture Notes in Computer Science*, pages 1–18, 2013.
19. I. Damgård and J. B. Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 247–264, 2003.
20. I. Damgård, V. Pastro, N.P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662, 2012.
21. I. Damgård and S. Zakarias. Constant-overhead secure computation for boolean circuits in the preprocessing model. In *TCC*, volume 7785 of *Lecture Notes in Computer Science*, pages 621–641, 2013.
22. C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.
23. C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178. ACM, 2009.
24. C. Gentry, S. Halevi, and N. P. Smart. Fully homomorphic encryption with polylog overhead. In *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 465–482, 2012.
25. C. Gentry, S. Halevi, and N. P. Smart. Homomorphic evaluation of the AES circuit. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867, 2012.
26. O. Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.
27. M. Hirt and J.B. Nielsen. Robust multiparty computation with linear communication complexity. In *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 463–482, 2006.
28. J.B. Nielsen, P.S. Nordholt, C. Orlandi, and S.S. Burra. A new approach to practical active-secure two-party computation. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 681–700, 2012.
29. N. P. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *PKC*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443, 2010.
30. N.P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. *To Appear in Designs, Codes and Cryptography*, 2012.