On the Communication required for Unconditionally Secure Multiplication

Ivan Damgård, Jesper Buus Nielsen, and Antigoni Polychroniadou

Dept. of Computer Science, Aarhus University.
{ivan,jbn,antigoni}@cs.au.dk

Abstract. Many information theoretically secure protocols are known for general secure multi-party computation, both in the honest majority setting, and in the dishonest majority setting with preprocessing. All known protocols that are efficient in the circuit size of the evaluated function follow the same *typical* "gate-by-gate" design pattern: we work our way through a boolean or arithmetic circuit, maintaining as an invariant that after we process a gate, the output of the gate is represented as a random secret sharing among the players. Finally, all shares for the outputs are revealed. This approach usually allows non-interactive processing of addition gates but requires communication for every multiplication gate. This means that while information theoretically secure protocols are very efficient in terms of computational work, they (seem to) require more communication and more rounds than computationally secure protocols. Whether this is inherent is an open and probably very hard problem. However, in this work we show that it is indeed inherent for protocols that follow the "gate by gate" design pattern. In particular, we present the following results:

- In the honest majority setting, any gate-by-gate protocol must communicate for every multiplication gate, even if only semi-honest security is required.
- For dishonest majority with preprocessing, a different proof technique is needed. We again show that any gate-by-gate protocol must communicate for every multiplication gate when the underlying secret sharing scheme is the additive one. We obtain similar results for arbitrary secret sharing schemes.
- In the honest majority setting, we also show that amortising over several multiplication gates can at best save an $\mathcal{O}(n)$ factor on the computational work.

All our lower bounds are met up to a constant factor by known protocols that follow the typical gate-by-gate paradigm. Our results imply that a fundamentally new approach must be found in order to improve the communication complexity of known protocols that are efficient in the circuit size of the function, such as GMW, SPDZ etc.

1 Introduction

Secure Multi-Party Computation (MPC) allows n players to compute an agreed function on privately held inputs, such that the desired result is correctly com-

puted and is the only new information released. This should hold, even if t players have been actively or passively corrupted by an adversary.

If point-to-point secure channels between players are assumed, any function can be computed with unconditional (perfect) security, against a passive adversary if $n \geq 2t+1$ and against an active adversary if $n \geq 3t+1$[BGW88,CCD88]. If we assume a broadcast channel and accept a small error probability, $n \geq 2t+1$ is sufficient to get active security[RBO89].

The protocols behind these results require a number of communication rounds that is proportional to the depth of an (arithmetic) circuit computing the function. Moreover, the communication complexity is proportional to the size of the circuit. Whether we can have constant round protocols and/or communication complexity much smaller than the size of the circuit and still be efficient (polynomial-time) in the circuit size of the function is a long-standing open problem. Note that this is indeed possible if one makes computational assumptions. Note also that if we give up on being efficient in the circuit size, then there are unconditionally secure and constant round protocols for any function [IK00] (which will, however, be very inefficient in general with respect to the computation). Moreover, there are works that apply to special classes of circuits (e.g., constant-depth circuits [BI05]) or protocols that require exponential amount of computation [BFKR90,NN01] and exponential storage complexity [IKM$^+$13]. Furthermore, the authors in [CK93] prove tight lower bounds for secure addition and the work of [DPP14] study the communication complexity of information theoretic protocols in a specific three-party model where two parties have input and a third gets the output.

The above issues are not only of theoretical interest: the methods we typically use in information theoretically secure protocols tend to be computationally much more efficient than the cryptographic machinery we need for computational security. So unconditionally secure protocols are very attractive from a practical point of view, except for the fact that they seem to require a lot of interaction.

The fact that existing information theoretically secure protocols (which are efficient in the circuit size of the function) have large round and communication complexity is a natural consequence of the fact that all such protocols follow the same *typical* "gate-by-gate" design pattern: Initially all inputs are secret-shared among the players. Now, for each gate in the circuit, where both its inputs have been secret-shared, we execute a subprotocol that produces the output from the gate in a secret-shared form. The protocol maintains as an invariant that for all gates that have been processed so far, the secret-sharing of the output value is of the same form used for the inputs (so we can continue processing gates) and is appropriately randomised such that one could open this sharing while revealing only that output value. As a result, it is secure to open the final outputs from the circuit.

For all known constructions which are efficient in the circuit size of the function, it is the case that multiplication gates require communication to be processed (while linear gates usually do not). It therefore follows that the number of rounds is at least the (multiplicative) depth of the circuit, and the communi-

cation complexity is $\Omega(ns)$ for a circuit of size s (the size being measured as the number of multiplication gates) in the worst case for $t < n/3$ and $t < n/2$ see the results of [DN07] and [BSFO12,GIP+14,GIP15], respectively. Note that protocols that tolerate a sub-optimal number of corrupted parties (e.g., $t < 0.49n$) and are based on packed secret-sharing techniques can reduce the amortised cost of multiplications if they can be parallelised [DIK+08,IPS09,DIK10,GIP15]. These techniques do not apply to all circuits, in particular not to "tall and skinny" circuits whose multiplicative depth is comparable to their size. In addition, they can at best save an $\mathcal{O}(n)$ factor in communication and computational work.

The situation is essentially the same for recent protocols that are designed for dishonest majority in the preprocessing model [DPSZ12,NNOB12] (except that amortization based on packed secret-sharing does not apply here due to the dishonest majority setting).

1.1 Contributions

In this paper, we ask a very natural question for unconditionally secure protocols which, to the best of our knowledge, has not been studied in detail before:

Is it really inherent that the typical *gate-by-gate approach to secure computation requires communication for each multiplication operation?*

Our Model. To avoid misunderstandings, let us be more precise about the model we assume: we consider synchronous protocols that are semi-honest and statistically secure against static corruption of at most t of the n players. We assume that point-to-point secure channels are available, and protocols are allowed to have dynamic communication patterns (in a certain sense we make precise later), i.e., it is not fixed a priori whether a protocol sends a message in a given time slot. Moreover, there is no bound on the computational complexity of protocols, in particular arbitrary secret sharing schemes are allowed. A *gate-by-gate* protocol is a protocol that evaluates an arithmetic circuit and for every multiplication gate, it calls a certain type of subprotocol we call a *Multiplication gate protocol* (MGP). We define MGPs precisely later, but they basically take as input random shares of two values a, b from a field and output random shares of $c = ab$. Neither the MGP nor the involved secret sharing schemes have to be the same for all gates. We do not even assume that the same secret sharing scheme is used for the inputs and output of an MGP, we only require that the *threshold* for the output sharing be the same as for the input. An *ordered* gate-by-gate protocol must call the MGP's in an order corresponding to the order in which one would visit the gates when evaluating the circuit, whereas this is not required in general. Thus the gate-by-gate notion is somewhat more general than what one might intuitively expect and certainly includes much more than, say the standard BGW protocol – which, of course, makes our negative results stronger.

Note that if multiplications did not require communication, it would immediately follow (for semi-honest security) that we would have an unconditionally

secure two-round protocol for computing any function. But this is not a priori impossible: it follows, for instance, from [IK00,IKM+13], that if less than a third of the players are corrupted, there is indeed such a two-round protocol (which, however, requires super polynomial computational work in general).

Honest Majority Setting. For honest majority protocols it is relatively easy to show that multiplications do require communication: we argue in the paper that any MGP secure against t corruptions requires that at least $t+1$ players communicate. For protocols with dynamic communication pattern this bound holds in expectation. It turns out that a protocol beating this bound would imply an unconditionally secure two-party protocol computing a multiplication, which is well known to be impossible. This implies that the communication complexity of any gate-by-gate protocol for honest majority must be proportional to $n \cdot s$ where s is the circuit size and that the round complexity of an ordered gate-by-gate protocol must be at least proportional to the multiplicative depth of the circuit. This matches the best protocols we know for general circuits up to a constant factor.

A gate-by-gate protocol is not allowed to amortise over several multiplications that can be done in parallel. This is anyway not possible in general, for instance if we evaluate a worst-case "tall and skinny" circuit. But for more benign circuits this is indeed an option. However, we show that amortising over several multiplication gates can save at most an $\mathcal{O}(n)$ factor in the computational work, which matches what we can get from known techniques based on packed secret-sharing. It is open if a similar bound holds for communication. This bound is a bit more tricky to prove than the first result. We based it on a lower bound by Winkler and Wullschleger [WW10] on the amount of preprocessed data one needs for (statistically) secure two-party computation of certain functions. It is perhaps somewhat surprising that an information theoretic bound on size of data translates to a bound on local computation.

Dishonest Majority Setting with Preproccesing. The argument used for the honest majority case breaks down if we consider protocols in the preprocessing model: here it is indeed possible to compute multiplications with unconditional security, even if $t = n - 1$ of the n players are corrupt. Nevertheless, we show similar results for this setting: here, any MGP secure against $t = n - 1$ corruptions must have $\Omega(n)$ players communicate. Note that existing constructions [DPSZ12] meet the resulting bound for gate-by-gate protocols up to a constant factor.

To obtain the result, we exploit again the lower bound by Winkler and Wullschleger, but in a different way. In a nutshell, we show that constructions beating our bound would imply a protocol that is too good to be true according to [WW10].

The result holds exactly as stated above assuming that the target secret-sharing scheme that the protocol outputs shares in is of a certain type that includes the simple additive secret-sharing scheme (which is also used in [DPSZ12],

[NNOB12]). If we put no restrictions on the target scheme, the results get a bit more complicated. Essentially what we show is the following: suppose we are given secret-sharings of inputs x_1, x_2 and want to compute shares in a function value $g(x_1, x_2)$. This is a generalisation of the case of multiplication where x_1, x_2 are field elements and $g(x_1, x_2) = x_1 x_2$. Now, if computing g securely for two players requires a large enough amount of preprocessed data, then we show that all n players must communicate to produce a random secret-sharing of $g(x_1, x_2)$, no matter what target secret sharing scheme is used. An example of such a function is the inner product where the input is a pair of m-vectors for large enough m and g outputs the inner product. It is the target secret-sharing scheme that determines what "large enough" means, see more details within.

2 Preliminaries

Notation. Let $[n]$ denote the set $\{1, 2, ..., n\}$. Calligraphic letters denote sets. The distribution of a random variable X over $\mathcal{X}$ is denoted by P_X. Given the distribution P_{XY} over $\mathcal{X} \times \mathcal{Y}$, the marginal distribution is denoted by $P_X(x) := \sum_{y \in \mathcal{Y}} P_{XY}(x, y)$. A conditional distribution $P_{X|Y}(x, y)$ over $\mathcal{X} \times \mathcal{Y}$ defines for every $y \in \mathcal{Y}$ a distribution $P_{X|Y=y}$. The conditional Shannon entropy of X given Y is defined as $H(X|Y) := -\sum_{x,y} P_{XY}(x, y) \log P_{X|Y}(x, y)$ where all logarithms are binary. Moreover, we use $h(p) = -p \log p - (1 - p) \log(1 - p)$ for the binary entropy function.

Protocols. We consider protocols involving n parties, denoted by $\mathcal{P} = \{P_1, \ldots, P_n\}$. The parties communicate over synchronous, point-to-point secure channels. We consider non-reactive secure computation tasks, defined by a deterministic or randomized functionality $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_n \to \mathcal{Z}_1 \times \ldots \times \mathcal{Z}_n$. The functionality specifies a mapping from n inputs to n outputs which the parties want to compute. The functionality can be fully specified by a conditional probability distribution $P_{Z_1 \cdots Z_n | X_1 \cdots X_n}$, where X_i is a random variable over $\mathcal{X}_i$, Z_i is a random variable over $\mathcal{Z}_i$, and for all inputs $(x_1, \ldots, x_n)$ we have a probability function $P_{Z_1 \cdots Z_n | X_1 \cdots X_n = (x_1, \ldots, x_n)}$ and $P_{Z_1 \cdots Z_n | X_1 \cdots X_n = (x_1, \ldots, x_n)}(z_1, \ldots, z_n)$ is the probability that the output is $(z_1, \ldots, z_n)$ when the input is $(x_1, \ldots, x_n)$. Vice versa, we can consider any conditional probability distribution $P_{Z_1 \cdots Z_n | X_1 \cdots X_n}$ as a specification of a probabilistic functionality. In the following we will freely switch between the terminology of probabilistic functionalities and conditional probability distributions.

In the sequel we consider the following two-party functionalities.

Definition 1 (Multiplication MULT functionality). *Let $\mathbb{F}$ be a finite field. Consider two parties A and B. We define the two-party functionality* $\mathrm{MULT}(a, b)$ *which on input $a \in \mathbb{F}$ from party A and $b \in \mathbb{F}$ from party B outputs* $\mathrm{MULT}(a, b) = a \cdot b$ *to both parties.*

Definition 2 (Inner Product $\mathbf{IP}_\kappa$ functionality). *Let $\mathbb{F}$ be a finite field and let $\kappa \geq 1$. Consider two parties A and B. We define the two-party functionality*

$\mathrm{IP}_\kappa(a,b)$ *which on input* $a \in \mathbb{F}^\kappa$ *from party* A *and* $b \in \mathbb{F}^\kappa$ *from party* B *outputs* $\mathrm{IP}_\kappa(a,b) = \sum_{i \leq \kappa} a_i b_i$ *to both parties.*

We consider stand-alone security. We consider static and passive corruptions of t out of n parties for some t. This means that a set of t parties are announced to be corrupted before the protocol is executed, and the corrupted parties still follow the protocol but might pool their views of the protocol to learn more than they should. We consider statistical correctness and statistical security. We allow simulators to be inefficient. Except that we do not consider computational security, the above model choices are the possible weakest ones, which just makes our impossibility proofs stronger.

The Security Parameter. The security is measured in a security parameter σ and we require that the "insecurity" goes to 0 as σ grows. We do not allow n to grow with σ, i.e., we require that the protocol can be made arbitrarily secure when run among a fixed set of parties by just increasing σ. The literature sometimes consider protocol which only become secure when run among a sufficiently large number of parties. We do not cover such protocols.

Communication Model. We assume that each pair of parties are connected by a secure communication channel, which only leaks to the adversary the length of each message sent[1]. We consider protocols proceeding in synchronous rounds. We consider a notion of communication complexity which we call the *Anticipated Communication Model*: In each round each party will first specify for each other party how long the message it wants to receive from that party is in this round. Length 0 means that it does not expect a message from that party in the given round. Then it will specify which other parties it wants to send messages to and will specify what these messages are. Then the messages are exchanged. If the length of a sent message does not match the length specified by the receiver the receiver will terminate with an error symbol $\perp$ as output, which will make it count as a violation of correctness.

Definition 3 (Anticipated message complexity). *We will say that the expected* anticipated message complexity *is the expected number of times a player sends or anticipates a non-empty message. The expectation is taken over the randomness of the players and maximised over all inputs.*

Note that if the anticipated message complexity is T this implies that in expectation at least $T/2$ messages are sent and hence at least $T/2$ bits are communicated. The reason for this slightly technical notion is to avoid a problem we would have if we allowed the communication pattern to vary arbitrarily: Consider a setting where P_j wants to send a bit b to P_i. If $b = 0$ it sends no message to P_i or say the empty string. If $b = 1$ it sends 0 to P_i. If b is uniformly random, then in half the cases P_j sends a message of length 0 and in half the cases it sends

[1] This is a standard way to model secure communication by an ideal functionality since any implementation using crypto would leak the message length.

a message of length 1. This means that a more liberal way of counting the communication complexity would say that the expected communication complexity is $\frac{1}{2}$. This would allow to exchange 1 bit of information with an expected $\frac{1}{2}$ bits of communication. This does not seem quite reasonable. The anticipated model avoids this while still allowing the protocol to have a dynamic communication pattern. Note that since we want to prove impossibility it is stronger to allow protocols with dynamic rather than fixed communication patterns.

Protocols with Preprocessing. We will also consider protocol for the preprocessing model. In the preprocessing model, the specification of a protocol also includes a joint distribution $P_{R_1 \cdots R_n}$ over $\mathcal{R}_1 \times \ldots \times \mathcal{R}_n$, where the $\mathcal{R}_i$'s are finite randomness domains. This distribution is used for sampling correlated random inputs $(r_1, \ldots, r_n) \leftarrow P_{R_1 \cdots R_n}$ received by the parties before the execution of the protocol. Therefore, the preprocessing is independent of the inputs. The actions of a party P_i in a given round may in this case depend on the private random input r_i received by P_i from the distribution $P_{R_1 \cdots R_n}$ and on its input x_i and the messages received in previous rounds. In addition, the action might depend on the statistical security paramenter σ which is given as input to all parties along with x_i and r_i. Using the standard terminology of secure computation, the preprocessing model can be thought of as a hybrid model where the parties have one-time access to an ideal randomized functionality P (with no inputs) providing them with correlated, private random inputs r_i.

Security Definition. A protocol securely implements an ideal functionality with an error of ϵ, if the entire view of each player can be simulated with an error of at most ϵ in an ideal setting, where the players only have black-box access to the ideal functionality.

Definition 4. *Let Π be a protocol for the $P_{R_1 \cdots R_n}$-preprocessing model. Let $P_{Z_1 \cdots Z_n | X_1 \cdots X_n}$ be an n-party functionality. Let $\mathcal{A} \subseteq \{1, \ldots, n\}$. Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$ be an input. Let $\mathsf{View}_{\mathcal{A}}^{\Pi}(\sigma, \boldsymbol{x})$ be the view of the parties P_i for $i \in \mathcal{A}$ in a random run of the protocol Π on input $\boldsymbol{x}$ and with security parameter σ. Let $\mathsf{Pattern}^{\Pi}(\sigma, \boldsymbol{x})$ denote the communication pattern in the same random run of the protocol Π, i.e., the list of the length of the messages exchanged between all pairs of parties in all rounds. Let $\mathsf{Output}_{\mathcal{A}}^{\Pi}(\sigma, \boldsymbol{x})$ be just the inputs and outputs of the parties P_i for $i \notin \mathcal{A}$ in the same random run of the protocol Π. Let*

$$\mathsf{Exec}_{\mathcal{A}}^{\Pi}(\sigma, \boldsymbol{x}) = (\mathsf{View}_{\mathcal{A}}^{\Pi}(\sigma, \boldsymbol{x}), \mathsf{Pattern}^{\Pi}(\sigma, \boldsymbol{x}), \mathsf{Output}_{\mathcal{A}}^{\Pi}(\sigma, \boldsymbol{x})) \ .$$

Let S be a randomized function called the simulator. *Sample $\boldsymbol{z}$ according to $P_{Z_1 \cdots Z_n | X_1 \cdots X_n}(\boldsymbol{x})$. Give input $\{(x_i, z_i)\}_{i \in \mathcal{A}}$ to S let $S(\{(x_i, z_i)\}_{i \in \mathcal{A}})$ denote the random variable describing the output of S. Let*

$$\mathsf{Sim}_S(\sigma, \boldsymbol{x}) = (S(\{(x_i, z_i)\}_{i \in \mathcal{A}}, \{(x_i, z_i)\}_{i \notin \mathcal{A}}) \ .$$

Let $\epsilon : \mathbb{N} \to [0, 1]$. Let SD *denote statistical distance. The protocol is ϵ-semi-honest secure with threshold t if there exist S such that for all $\boldsymbol{x}$ and all $\mathcal{A}$ with*

$|\mathcal{A}| \leq t$ *it holds that*

$$\mathrm{SD}(\mathsf{Exec}^{\Pi}_{\mathcal{A}}(\sigma, \boldsymbol{x}), \mathsf{Sim}_S(\sigma, \boldsymbol{x})) \leq \epsilon(\sigma) \ .$$

We say that ϵ is negligible if for all $c \in \mathbb{N}$ there exists $\sigma_c \in \mathbb{N}$ such that $\epsilon(\sigma) \leq \sigma^{-c}$ for all $\sigma \geq \sigma_c$. The protocol is statistically semi-honest secure with threshold t if it is ϵ-semi-honest secure for a negligible ϵ.

Secret-Sharing. A $(t+1)$-out-of-n secret-sharing scheme takes as input a secret s from some input domain and outputs n shares, with the property that it is possible to efficiently reconstruct s from every subset of $t+1$ shares, but every subset of at most t shares reveals nothing about the secret s. The value t is called the privacy *threshold* of the scheme.

A secret-sharing scheme consists of two algorithms: the first algorithm, called the *sharing algorithm* Share, takes as input the secret s and the parameters t and n, and outputs n shares. The second algorithm, called the *recovery algorithm* $\mathsf{Recover}$, takes as input $t+1$ shares and outputs a value s. It is required that the reconstruction of shares generated from a value s produces the same value s. Formally, consider Definition 5 below.

Definition 5 (Secret-sharing). *Let $\mathbb{F}$ be a finite field and let $n, t \in \mathbb{N}$. A pair of algorithms $\mathcal{S}^n_t = (\mathsf{Share}, \mathsf{Recover})$ where Share is randomized and $\mathsf{Recover}$ is deterministic are said to be a secret-sharing scheme if for every $n, t \in \mathbb{N}$, the following conditions hold.*

Reconstruction: *For any set $\mathcal{T} \subseteq \mathcal{P}$ such that $|\mathcal{T}| > t$ and for any $s \in \mathbb{F}$ it holds that*

$$\Pr[\mathsf{Recover}(\mathsf{Share}_{\mathcal{T}}(s, n, t)) = s] = 1$$

where $\mathsf{Share}_{\mathcal{T}}$ is the restriction of the outputs of Share to the elements in $\mathcal{T}$.

Privacy: *For any set $\mathcal{T} \subseteq \mathcal{P}$ such that $|\mathcal{T}| \leq t$ and for any $s, s' \in \mathbb{F}$ it holds that*

$$\mathsf{Share}_{\mathcal{T}}(s, n, t) \equiv \mathsf{Share}_{\mathcal{T}}(s', n, t)$$

where we use $\equiv$ to denote that two random variables have the same distribution.

Additive Secret-Sharing In an additive secret-sharing scheme, n parties hold shares the sum of which yields the desired secret. By setting all but a single share to be a random field element, we ensure that any subset of $n-1$ parties cannot recover the initial secret.

Definition 6 (Additive secret-sharing). *Let $\mathbb{F}$ be a finite field and let $n \in \mathbb{N}$. Consider the secret-sharing scheme $\mathcal{A}^n = (\mathsf{Share}, \mathsf{Recover})$ defined below.*

- *The algorithm Share on input (s, n) performs the following:*
  1. *Generate $(s_1, \ldots, s_{n-1})$ uniformly at random from $\mathbb{F}$ and define $s_n = s - \sum_{i=1}^{n-1} s_i$.*

2. Output $(s_1, \ldots, s_n)$ *where* s_i *is the share of the* i*-th party.*
- *The recovery algorithm* Recover *on input* $(s_1, \cdots, s_n)$*, outputs* $\sum_{i=1}^{n} s_i$*.*

It is easy to show that the distribution of any $n-1$ of the shares is the uniform one on $\mathbb{F}^{n-1}$ and hence independent of s.

More Notation. In the sequel for a value $s \in \mathbb{F}$ we denote by $[s]^{\mathcal{S}_t^n}$ a random sharing of s for the secret-sharing scheme $\mathcal{S}_t^n$. That is, $[s]^{\mathcal{S}_t^n} \leftarrow \mathsf{Share}(s, n, t)$ where $[s]^{\mathcal{S}_t^n} = (s_1, \ldots, s_n)$. Similarly, we denote by $[s]^{\mathcal{A}^n}$ a random additive sharing of s secret shared among n parties. Moreover, we denote by Π_f an n-party protocol for a function f and by $\Pi_f^{A,B}$ a two-party protocol between parties A and B.

3 Secure Computation in the Plain Model

We first investigate the honest majority scenario. As explained in the introduction, we will consider protocols that compute arithmetic circuits over some field securely using secret-sharing. All known protocols of this type handle multiplication gates by running a subprotocol that takes as input shares in the two inputs a and b to the gate and output shares of the product ab, such that the output shares contain only information about ab (and no side information on a nor b). Accordingly, we define below a *multiplication gate protocol* (MGP) to be an interactive protocol for n players that does exactly this, and then show a lower bound on the communication required for such a protocol.

Definition 7 (Multiplication Gate Protocol $\Pi_{\mathbf{MULT}}$). *Let* $\mathbb{F}$ *be a finite field and let* $n \in \mathbb{N}$*. Let* $\mathcal{S}_t^n$ *and* $\hat{\mathcal{S}}_t^n$ *be two secret-sharing schemes as per Definition 5. A protocol* Π_{MULT} *is a* Multiplication Gate Protocol (MGP) *with threshold* t*, input sharing-scheme* $\mathcal{S}_t^n$ *and output sharing-scheme* $\hat{\mathcal{S}}_t^n$ *if it satisfies the following properties:*

Correctness: *In the interactive protocol* Π_{MULT}*, players start from sets of shares* $[a]^{\mathcal{S}_t^n} \leftarrow \mathsf{Share}(a, n, t)$ *and* $[b]^{\mathcal{S}_t^n} \leftarrow \mathsf{Share}(b, n, t)$*. Each player outputs a share such that these together form a set of shares* $[ab]^{\hat{\mathcal{S}}_t^n}$*.*

Simulation: *If the protocol is run on randomly sampled shares* $[a]^{\mathcal{S}_t^n} \leftarrow \mathsf{Share}(a, n, t)$ *and* $[b]^{\mathcal{S}_t^n} \leftarrow \mathsf{Share}(b, n, t)$*, then the only new information the output shares can reveal to the adversary is* ab*. More formally, for each player subset* $\mathcal{A}$ *of size at most* t*, there exists a simulator* $S_{\mathcal{A}}$ *which when given* $[ab]^{\hat{\mathcal{S}}_t^n}$ *and the shares of the parties* $\mathcal{A}$ *in* $[a]^{\mathcal{S}_t^n}$ *and* $[b]^{\mathcal{S}_t^n}$ *will output a view which is statistically indistinguishable from views of players in* $\mathcal{A}$ *in an execution of* $\Pi_{\mathrm{MULT}}([a]^{\mathcal{S}_t^n}, [b]^{\mathcal{S}_t^n})$*.*

Note that we do not require that the input and output sharing schemes to be the same, only that the threshold t is preserved. Also note that we do not require the simulators to be efficient.

Theorem 1. *There exists no* MGP Π_{MULT} *as per Definition 7 with threshold* t *and with expected anticipated message complexity* $\leq t$*.*

Proof. Suppose for contradiction that there exists a MGP Π_{MULT} with expected anticipated communication complexity at most t. We first do the proof in the simple case where the communication pattern is fixed. This means that at most t parties are communicating, i.e., they send or receive messages and the set of parties that communicate is known and fixed. We will assume for notational convenience that it is the parties $P_1, \ldots, P_t$. We are going to use Π_{MULT} to construct a two-party unconditionally secure protocol $\Pi^{A,B}_{\text{MULT}}$ which securely computes the MULT function as per Definition 1. Given two parties A and B involved in the $\Pi^{A,B}_{\text{MULT}}$ protocol, we are going to let A on input $a \in \mathbb{F}$ emulate the t parties that communicate and B on input $b \in \mathbb{F}$ emulates the rest of the parties but we are interest just for one additional party, say P_{t+1}. In particular, protocol $\Pi^{A,B}_{\text{MULT}}$ proceeds as follows:

Protocol $\Pi^{A,B}_{\text{MULT}}(a, b)$

Input Phase

Parties A, B secret share their inputs a, b using the secret-sharing scheme $\mathcal{S}^n_t$. More specifically, A computes $[a]^{\mathcal{S}^n_t} \leftarrow \mathsf{Share}(a, n, t)$ and B computes $[b]^{\mathcal{S}^n_t} \leftarrow \mathsf{Share}(b, n, t)$.

Party A sends the input shares $(a_{t+1}, \ldots, a_n)$ to party B and Party B sends the input shares $(b_1, \ldots, b_t)$ to party A.

Evaluation Phase

Parties A, B invoke the protocol $\Pi^{\mathcal{S}^n_t}_{\text{MULT}}(a_1, \ldots, a_{t+1} \ldots a_n, b_1, \ldots, b_{t+1}, \ldots b_n)$. The emulation of $\Pi^{\mathcal{S}^n_t}_{\text{MULT}}$ yields a sharing of $[c]^{\hat{\mathcal{S}}^n_t}$ and outputs $(c_1, \ldots, c_t)$ to party A and $(c_{t+1}, \ldots, c_n)$ to party B.

Output Phase

Party A sends the output shares $(c_1, \ldots, c_t)$ to party B and Party B sends the output share (c_{t+1}) to party A.

Each party given $t + 1$ shares of c recovers the output $c = a \cdot b$

We now show that the above protocol is correct and secure. Correctness follows immediately. The protocol is secure(private) because of the simulation property of Π_{MULT}. More precisely, since A emulates the parties that communicate in Π_{MULT}, the 2-party protocol requires communication only in the input and the output phases. The shares received by a player in the input phase can be simulated by the privacy property of the input sharing scheme, and the shares received in the output phase can be simulated by invoking the simulator guaranteed by Definition 7. However, the above leads to a contradiction since it is well known [BGW88,CCD88] that it is impossible to realize passively secure two-party multiplication in the information theoretic setting (even if inefficient simulators are allowed). The theorem follows.

We now address the case where the communication pattern might be dynamic. We say that a party communicated if it sent a non-empty message or if it anticipated a non-empty message. So by definition, the expected number of communicating parties is $\leq t/2$, and the observed value is at most its expectation with at least some constant probability p (since it is non-negative). Hence with probability at least p, at most t parties communicate. We can therefore pick

a subset $\mathcal{C}$ of the parties of size t such that it happens with probability at least $p/\binom{n}{t}$ that only the parties in $\mathcal{C}$ communicate. Since we can increase the security paramenter σ independently of n, the number $p/\binom{n}{t}$ is a positive constant (in σ). We can then modify $\Pi^{A,B}_{\text{MULT}}(a,b)$ such that B runs the parties in $\mathcal{C}$ and A runs some other party. The protocol runs as $\Pi^{A,B}_{\text{MULT}}(a,b)$ except that if it happens, while B runs the parties in $\mathcal{C}$, that a party in $\mathcal{C}$ anticipates a non-empty message from a party outside $\mathcal{C}$, then B terminates the execution and sends A a message to that effect. The protocol is likewise terminated if the parties that A simulates anticipate any message in any round. In case the protocol terminates, the two parties just try again. Since $p/\binom{n}{t}$ is a positive constant this succeeds in an expected constant number of tries. Notice that when the protocol succeeds, all parties in $\mathcal{C}$ received all the messages they would have received in a run of $\Pi^{A,B}_{\text{MULT}}(a,b)$ where all the parties were active, as parties only receive the messages they anticipate. Hence the parties in $\mathcal{C}$ have correct outputs (except with negligible probability). For the same reason the output of the party simulated by B will be correct. Hence A and B can reconstruct the output from these $t+1$ shares. We can also argue that the protocol is private: since we assumed that the communication pattern is leaked to the adversary it cannot depend on the inputs and can therefore be simulated with statistically close distribution. This allows us to simulate all the runs of the MGP, while the last part can be simulated as we did before. □

In the following, we will use the term *gate-by-gate* protocol to refer to any protocol that computes an arithmetic circuit securely by invoking an MGP for each multiplication gate in the circuit such that the sets of shares that are input are randomly chosen. We leave unspecified what happens with addition gates as this is irrelevant for the bounds we show. An *ordered* gate-by-gate protocol invokes MGP's for multiplication gates in an order corresponding to the order in which one would visit the gates when evaluating the circuit.

The implication of the above theorem is that any gate-by-gate protocol that is secure against $t = \Theta(n)$ corruptions must communicate $\Omega(n \cdot |C|)$ bits where $|C|$ is the size of the circuit, and moreover, an ordered gate-by-gate protocol must have a number of rounds that is proportional to the (multiplicative) depth of the circuit.

Jumping ahead, we note that the arguments for this conclusion break down completely when we consider secure computation in the preprocessing model with dishonest majority since here it is no longer true that two-party unconditionally secure multiplication is impossible: just a single preprocessed multiplication triple will be enough to compute a multiplication. We return to this issue in the next section.

Amortised Multiplication Gate Protocols. There is one clear possibility for circumventing the bounds we just argued for gate-by-gate protocols, namely: what if the circuit structure allows us to do, say k multiplications in parallel? Perhaps this can be done more efficiently than k separate multiplications? Of course, this will not help for a worst case circuit whose depth is comparable to

its size. But in fact, for "nicer" circuits, we know that such optimizations are possible, based on so-called packed secret-sharing. The catch, however, is that apart from loosing in resilience this only works if there is a gap of size $\Theta(k)$ between the privacy and reconstruction thresholds of the secret-sharing scheme used, so therefore the number of players must grow with k.

One may ask if this is inherent, i.e., can we save on the *communication* needed for many multiplication gates in parallel, only by increasing the number of players? While we believe this is true, we were not able to show this. But we were able to show a similar result for *computational* complexity, as detailed below.

First, we can trivially extend Definition 5 to cover schemes in which the secret is a vector $(a_1, ..., a_k)$ of field elements instead of a single value. A further extension covers *ramp* schemes in which there are two thresholds: the privacy threshold t which is defined as in Definition 5 and a reconstruction threshold $r > t$, where any set of size at least r can reconstruct the secret. Such a scheme is denoted by $\mathcal{S}^n_{t,r}$. Note that the shares in this case may be shorter than the secret, perhaps even a single field element per player.

We can now define a simple extension of the multiplication gate protocol concept:

Definition 8 (*k-Multiplication Gate Protocol* $\Pi_{\mathbf{MULT}^k}$)**.** *Let* $\mathbb{F}$ *be a finite field and let* $n \in \mathbb{N}$*. Let* $\mathcal{S}^n_{t,r}$ *and* $\hat{\mathcal{S}}^n_{t,r}$ *be two ramp sharing schemes defined over* $\mathbb{F}$*. A protocol* Π^k_{MULT} *is said to be a k-Multiplication Gate Protocol* (k-MGP) *with thresholds* t, r*, input sharing scheme* $\mathcal{S}^n_{t,r}$ *and output sharing scheme* $\hat{\mathcal{S}}^n_{t,r}$ *if it satisfies the following properties:*

Correctness: *In the interactive protocol* Π_{MULT^k}*, players start from sets of shares* $[(a_1, ..., a_k)]^{\mathcal{S}^n_{t,r}}$ *and* $[(b_1, ..., b_k)]^{\mathcal{S}^n_{t,r}}$*. Each player outputs a share such that these together form a set of shares* $[(a_1b_1, ..., a_kb_k)]^{\hat{\mathcal{S}}^n_{t,r}}$*.*

Simulation: *When the protocol is run on random secret sharings, then the only new information the output shares can reveal to the adversary is* $a_1b_1, ..., a_kb_k$*. More formally, for each player subset* $\mathcal{A}$ *of size at most* t*, there exists a simulator* $S_{\mathcal{A}}$ *which when given* $[(a_1b_1, \ldots, a_kb_k)]^{\hat{\mathcal{S}}^n_{t,r}}$ *and the shares of the parties* $\mathcal{A}$ *in random sharings* $[(a_1, ..., a_k)]^{\mathcal{S}^n_{t,r}}$ *and* $[(b_1, ..., b_k)]^{\mathcal{S}^n_{t,r}}$ *will output a value with distribution statistically close to the view of players in* $\mathcal{A}$ *of an execution of* $\Pi_{\mathrm{MULT}^k}([(a_1, ..., a_k)]^{\mathcal{S}^n_{t,r}}, [(b_1, ..., b_k)]^{\mathcal{S}^n_{t,r}})$*.*

Before giving our result on k-MGPs we note that for any interactive protocol, it is always possible to represent the total computation done by the players as an arithmetic circuit over a finite field $\mathbb{F}$ (arithmetic circuits can emulate Boolean circuit which can in turn emulate Turing machines). We can encode messages as field elements and represent sending of messages by wires between the parts of the circuit representing sender and receiver. For protocol Π, we refer to an algorithm outputting such a circuit as *an arithmetic representation of* Π *over* $\mathbb{F}$. Note that such a representation is not in general unique, but once we have chosen one, it makes sense to talk about, e.g., the number of multiplications done by a player in Π.

Theorem 2. *Let $t < r \leq n \in \mathbb{N}$. Also let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of parties. Assume that the* k-MGP Π_{MULT^k} *defined over $\mathbb{F}$ has thresholds t, r. Then for any arithmetic representation of Π_{MULT^k} over $\mathbb{F}$ and for each subset $\mathcal{S} \subset \mathcal{P}$ of size $n - 2t$, the total number of multiplications done by players in $\mathcal{S}$ is $\Omega(k)$.*

Proof. Suppose for contradiction that there exists a k-MGP Π_{MULT^k} in which the total number of multiplications done by players in $\mathcal{S}$ is $o(k)$. Assume for notational convenience that $\mathcal{S} = \{P_{2t+1}, \ldots, P_n\}$. We are going to use it to construct a two-party unconditionally secure protocol $\Pi^{A,B}_{\text{MULT}}$ in the preprocessing model which securely computes k multiplications as follows. We let $u \leftarrow U$ denote the correlated randomness we will use in $\Pi^{A,B}_{\text{MULT}}$. Given two parties A and B involved in the $\Pi^{A,B}_{\text{MULT}}$ protocol, the idea is to use the assumed k-MGP where A emulates t players and B emulates another t players. In addition, parties A, B together emulate the rest of the parties in $\mathcal{S}$. This can be done using the preprocessed data u: we consider the parties in $\mathcal{S}$ as a reactive functionality $f_{\mathcal{S}}$ which can be implemented using an existing protocol in the preprocessing model. One example of such a protocol is the SPDZ protocol [DPSZ12] denoted by $\Pi^{SPDZ}_{f_{\mathcal{S}}}$ [2] which uses additive secret sharing. Therefore, protocol $\Pi^{A,B}_{\text{MULT}}$ proceeds as follows:

Protocol $\Pi^{A,B}_{\text{MULT}}(\{a_i\}_{i\in[k]}, \{b_i\}_{i\in[k]}, u)$:

Input Phase

$\forall i \in [k]$, parties A, B secret share their inputs a_i, b_i using the ramp sharing scheme $\mathcal{S}^n_{t,r}$. So A computes $[a_1, \ldots, a_k]^{\mathcal{S}^n_{t,r}} \leftarrow \mathsf{Share}((a_1, \ldots, a_k), n, t)$ and B computes $[b_1, \ldots, b_k]^{\mathcal{S}^n_{t,r}} \leftarrow \mathsf{Share}((b_1, \ldots, b_k), n, t)$. For simplicity of exposition, we denote by $(\bar{a}_1, \ldots, \bar{a}_n), (\bar{b}_1, \ldots, \bar{b}_n)$ the shares of $[a_1, \ldots, a_k]^{\mathcal{S}^n_{t,r}}$ and $[b_1, \ldots, b_k]^{\mathcal{S}^n_{t,r}}$, respectively.

Party A sends the input shares $(\bar{a}_1, \ldots, \bar{a}_t)$ to party B and Party B sends the input shares $(\bar{b}_t, \ldots, \bar{b}_{2t})$ to party A.

Additively secret share the inputs $(\bar{a}_{2t+1}, \ldots, \bar{a}_n, \bar{b}_{2t+1}, \ldots, \bar{b}_n)$ of the parties in $\mathcal{S}$ between A and B using the additive secret-sharing $\mathcal{A}^2$ and obtain the shares $([\bar{a}_{2t+1}]^{\mathcal{A}^2}, \ldots, [\bar{a}_n]^{\mathcal{A}^2}, [\bar{b}_{2t+1}]^{\mathcal{A}^2}, \ldots, [\bar{b}_n]^{\mathcal{A}^2})$. For the following phase, as we mentioned above, we will think of the computation done by the parties in $\mathcal{S}$ as a reactive functionality $f_{\mathcal{S}}$ which is implemented using the protocol $\Pi^{SPDZ}_{f_{\mathcal{S}}}$ in the preprocessing model.

Evaluation Phase

Parties A, B invoke the protocol $\Pi_{\text{MULT}^k}([a_1, \ldots, a_k]^{\mathcal{S}^n_{t,r}}, [b_1, \ldots, b_k]^{\mathcal{S}^n_{t,r}})$ in which A, B emulates t parties each, and they emulate the rest, $n - 2t$ players, using the preprocessed data u and $\Pi^{SPDZ}_{f_{\mathcal{S}}}$. To this end, note that $\Pi^{SPDZ}_{f_{\mathcal{S}}}$ represents data by additive secret-sharing. Values $(\bar{a}_{2t+1}, \ldots, \bar{a}_n, \bar{b}_{2t+1}, \ldots, \bar{b}_n)$ of the parties in $\mathcal{S}$ were already additively shared, so they can be used directly as input to $\Pi^{SPDZ}_{f_{\mathcal{S}}}$.

[2] We do passive security here, so a simpler variant of SPDZ will suffice, without authentication codes on the shared values.

Now, the emulation of Π_{MULT^k} is augmented with the protocol $\Pi_{f_{\mathcal{S}}}^{SPDZ}$ as follows: when a party in $\mathcal{S}$ would do a local operation, we do the same operation in $\Pi_{f_{\mathcal{S}}}^{SPDZ}$. When a party outside $\mathcal{S}$ sends a message to a party in $\mathcal{S}$ an additive secret-sharing of that message is formed between A and B. When a party in $\mathcal{S}$ sends a message to a party outside $\mathcal{S}$ the corresponding additive secret-sharing is reconstructed towards A or B, depending on who emulates the receiver. In the end, we will obtain additive sharings between A and B of the outputs of parties in $\mathcal{S}$, namely $([\bar{c}_{2t+1}]^{\mathcal{A}^2}, \ldots, [\bar{c}_n]^{\mathcal{A}^2})$.

Output Phase

A sends the output shares $(\bar{c}_1, \ldots, \bar{c}_t)$ to B, B sends the output shares $(\bar{c}_{t+1}, \ldots, \bar{c}_{2t})$ to A, and A and B exchange their additive shares $([\bar{c}_{2t+1}]^{\mathcal{A}^2}, \ldots, [\bar{c}_n]^{\mathcal{A}^2})$ in order to recover $(\bar{c}_{2t+1}, \ldots, \bar{c}_n)$.

Now both A and B have $n \geq r$ shares of the output and can recover the results $c_i = a_i \cdot b_i$ for all $i \in [k]$.

We now show that the above protocol is correct and secure. Correctness follows immediately from correctness of Π_{MULT^k} and $\Pi_{f_{\mathcal{S}}}^{SPDZ}$. The protocol is secure because $\Pi_{f_{\mathcal{S}}}^{SPDZ}$ is secure, and by the simulation property of the MGP Π_{MULT^k}. That is, the shares $(c_1, \ldots, c_r)$ under the secret-sharing scheme $\hat{\mathcal{S}}_{t,r}^n$, as computed in the final phase of $\Pi_{\text{MULT}}^{A,B}$, only reveal information about the outputs $\{a_i \cdot b_i\}_{i \in [k]}$. Note that $\Pi_{f_{\mathcal{S}}}^{SPDZ}$ reveals the structure of the circuit for $f_{\mathcal{S}}$. This is secure as we assume that the parties in $\mathcal{S}$ are represented as known arithmetic circuits.

Now note that the preprocessed data required by the protocol $\Pi_{f_P}^{SPDZ}$ amount to a constant number of field elements for each multiplication done. Since n grows with k in our case this means we need $o(k)$ preprocessed field elements by assumption on Π_{MULT^k}.

However, the above leads to a contradiction since by results in [WW10], it is impossible for two parties to compute k multiplications with statistical security using preprocessed data of size $o(k)$ field elements. □

What this theorem shows is, for instance, that if we want each player to do only a constant number of local multiplications in a k-MGP, then n needs to be $\Omega(k)$. Since this is precisely what protocols based on packed sharing can achieve (see, e.g., [DIK+08]), the bound in the theorem is in this sense tight.

4 Secure Computation in the Preprocessing Model

It is well known that all functions can be computed with unconditional security in the setting where $n-1$ of the n players may be corrupted, and where the players are given correlated randomness, also known as preprocessed data, that does not have to depend on the function to be computed, nor on the inputs. Winkler and Wullschleger [WW10] proved lower bounds on the the amount of preprocessed

data needed to compute certain functions with statistical security where the bound depends on certain combinatorial properties of the target function.

All existing protocols for this setting that are efficient in the circuit size of the function work according to the gate-by-gate approach we encountered in the previous section. We can define (ordered) gate-by-gate protocols and multiplication gate protocols exactly as for the honest majority setting except that multiplication gate protocols are allowed to consume preprocessed data.

As before, we want to show that multiplication gate protocols require a certain amount of communication, but as mentioned before, we can no longer base ourselves on impossibility of unconditionally secure multiplication for two parties, since this is in fact possible in the preprocessing model. Instead, the contradiction will come from the known lower bounds on the size of the preprocessed data needed to compute certain functions.

4.1 Protocols based on Additive Secret-Sharing

Theorem 3. *Consider the preprocessing model where $n-1$ of the n players may be passively corrupted. In this setting, there exists no* MGP Π_{MULT} *with expected anticipated communication complexity $\leq n-1$ and with additive secret-sharing $\mathcal{A}^n$ as output sharing scheme.*

Proof. Suppose for contradiction that there exists Π_{MULT} which contradicts the claim of the theorem. Similar to Theorem 1 we will first assume a fixed communication pattern. Assume for notational convenience that only the parties $P_1, \ldots, P_{n-1}$ communicate. Given two parties A and B, we are going to construct a two-party protocol $\Pi^{A,B}_{\text{MULT}}$ which on input $a, b \in \mathbb{F}$ from A, B, respectively, securely computes ab. The idea is for A to emulate the $n-1$ players who communicate in Π_{MULT} while B emulates the last player. In particular, protocol $\Pi^{A,B}_{\text{MULT}}$ proceeds as follows:

Protocol $\Pi^{A,B}_{\text{MULT}}$

Input Phase

Parties A, B secret share their inputs a, b using the input secret-sharing scheme $\mathcal{A}^n$ of Π_{MULT}. More specifically, A computes $[a]^{\mathcal{A}^n} \leftarrow \mathsf{Share}(a, n, n-1)$ and B computes $[b]^{\mathcal{A}^n} \leftarrow \mathsf{Share}(b, n, n-1)$.

Party A sends the input share a_n to party B and Party B sends the input shares $(b_1, .., b_{n-1})$ to party A.

Evaluation Phase

Parties A, B invoke the protocol Π_{MULT} where A emulates the $n-1$ players who communicate, and we assume these are the first $n-1$ players. This means that this phase involves no communication between A and B, but it may consume some preprocessed data. The execution of Π_{MULT} yields a sharing of $[c]^{\mathcal{A}^n}$ and outputs $(c_1, ..., c_{n-1})$ to party A and c_n to party B.

Output Phase

A sends $\sum_{i=1}^{n-1} c_i$ to B and B sends c_n to A. The parties add the received values to recover the output $c = a \cdot b$.

Correctness of this protocol follows immediately. The protocol can be argued to be secure(private) as follows: in the input phase, the parties receive only an unqualified set of shares whose distribution can be simulated perfectly. There is no communication to be simulated in the evaluation phase. In the output phase, it is the case for both parties that the value received from the other party is trivial to simulate because it is determined from the party's own value and the result ab.

However, we can say even more: Let u be the preprocessed data that is consumed during the protocol (Π_{MULT} uses preprocessed data). Note that the communication in $\Pi^{A,B}_{\text{MULT}}$ is actually independent of u. This means that we could execute k instances of $\Pi^{A,B}_{\text{MULT}}$ to compute k products on independent inputs, *while reusing the preprocessed data* u. The correctness is clearly not affected by this and the simulation is done by simply doing k independent simulations as specified above. This works since the simulation is also independent of u.

However, this leads to a contradiction with the result of [WW10]: they showed that the amount of preprocessed data needed for a secure multiplication is at least some non-zero number of bits w. It also follows from [WW10] that if we want k multiplications on independently chosen inputs this requires kw bits. So if we consider a k large enough that $kw > u$, we have a contradiction and the theorem follows.

We now generalise to dynamic communication patterns. As in the proof of Theorem 1 we can find a party P_i such that with some constant positive probability p the party P_i does not send a message and no party anticipates a message from P_i. Assume without loss of generality that this is party P_n. Assume first that p is negligibly close to 1. In that case the parties can apply the above protocol unmodified. Consider then the case where p is not negligibly close to 1. We also have that p is not negligibly close to 0. Hence there is a non-negligible probability that P_n sends a message and a non-negligible probability that P_n does not send a message. The decision of P_n to communicate or not can depend only on four values:

- Its share a_n of a.
- Its share b_n of b.
- Its share u_n of the correlated randomness.
- Its private randomness, call it r_n.

This means that there exist a function $\varrho(a_n, b_n, u_n, r_n) \in \{0, 1\}$ such that P_n communicates iff $\varrho(a_n, b_n, u_n, r_n) = 1$. Observe that the decision can in fact not depend more than negligibly on a_n and b_n. If it did, this would leak information on these shares to the parties $P_1, \ldots, P_{n-1}$ which already know all the other shares. This would in turn leak information on a or b to the parties $P_1, \ldots, P_{n-1}$, which would contradict the simulatability property of the protocol. We can therefore without loss of generality assume that there exist a function $\varrho(u_n, r_n) \in \{0, 1\}$ such that P_n communicates iff $\varrho(u_n, r_n) = 1$.

Assume that with non-negligible probability over the choice of the u_n received by P_n it happens that the function $\varrho(u_n, r_n)$ depends non-negligibly on r_n, i.e.,

for a uniform r_n it happens with non-negligible probability that $\varrho(u_n, r_n) = 0$ and it also happens with non-negligible probability that $\varrho(u_n, r_n) = 1$. Since r_n is independent of the view of the parties $P_1, \ldots, P_{n-1}$, as it is the private randomness of P_n, it follows that the probability that one of the other parties anticipate a message from P_n is independent of whether $\varrho(u_n, r_n) = 0$ or $\varrho(u_n, r_n) = 1$. Hence it either happens with non-negligible probability that $\varrho(u_n, r_n) = 0$ and yet one of the other parties anticipate a message from P_n or it happens with non-negligible probability that $\varrho(u_n, r_n) = 1$ and yet none of the other parties anticipate a message from P_n. Both events contradicts the correctness of the protocol. We can therefore without loss of generality assume that there exist a function $\varrho(u_n) \in \{0, 1\}$ such that P_n communicates iff $\varrho(u_n) = 1$. By assumption we have that p is non-zero, so there exist some u_n such that $\varrho(u_n) = 0$. We can therefore condition the execution on the event $\varrho(u_n) = 0$. Let U be the distribution from which u is sampled. Consider then the random variable U' which is distributed as U under the condition that $\varrho(u_n) = 0$. We claim that if we run Π_{MULT} with U' instead of U then the protocol is still secure. Assuming that this claim is true, A and B can apply the above protocol, but simply use (Π_{MULT}, U') instead of (Π_{MULT}, U).

What remains is therefore only to argue that (Π_{MULT}, U') is secure. To simulate the protocol, run the simulator for (Π_{MULT}, U) until it outputs a simulated execution where P_n did not communicate. Call this simulator S'. Let E be the event that P_n does not communicate. Since it can be checked from just inspecting the view of the real execution of (Π_{MULT}, U) (or the simulation) whether E occurred, it follows that E occurs with the same probability in the real execution and the simulation (or at least probabilities which are negligible close) or we could use the occurrence of E to distinguish. Since E happens with a positive constant probability it then also follows that the real execution conditioned on E and the simulation condition on E are indistinguishable, or we could apply a distinguisher for the conditioned distributions when E occurs and otherwise make a random guess to distinguish the real execution of (Π_{MULT}, U) from its simulation. This shows that S' simulates (Π_{MULT}, U'). □

A generalisation. We note that Theorem 3 easily extends to any output secret sharing scheme with the following property: Given shares $c_1, ..., c_n$ of c, there is a function ϕ such that one can reconstruct c from $c_1, ..., c_{n-1}, \phi(c_n)$ and given c and $c_1, ..., c_{n-1}$ one can simulate $\phi(c_n)$ with statistically close distribution. The proof is the same as above except that in the output phase, B sends $\phi(c_n)$ to A, who computes c and sends it to B.

Theorem 3 shows, for instance, that the SPDZ protocol [DPSZ12] has optimal communication for the class of gate-by-gate protocols using additive secret-sharing: it sends $O(n)$ messages for each multiplication gate, and of course one needs to send $\Omega(n)$ messages if all n players are to communicate, as mandated in the theorem. Note also that in the dishonest majority setting, the privacy threshold of the secret-sharing scheme used has to be $n - 1$, so we cannot have a gap between the reconstruction and privacy thresholds, and so amortisation

tricks based on packed secret-sharing cannot be applied. We therefore do not consider any lower bounds for amortised MGP's.

4.2 Protocols based on any Secret-Sharing Scheme

Note that if we consider an MGP whose output sharing scheme is not the additive scheme, the proof of Theorem 3 does not work. This is because it is no longer clear that given your own share of the product and the result, the other party's share is determined. In particular, the distribution of the other share may depend on the preprocessed data we consume and so it is no longer clear that we can reuse the preprocessing.

The solution is to use an existing reconstruction protocol secure in the preprocessing model to securely reconstruct the output from the shares held locally by the two parties after the execution of the MGP protocol. This will mean that we can indeed reuse preprocessed data consumed by the MGP protocol itself however, we now consume new preprocessed data for every instance of the reconstruction protocol since this protocol requires communication. It turns out that if we use a variant of the MGP that computes, not just a product, but an inner product of long enough vectors, we can still obtain a contradiction. The fact that we show it for the inner product is because we can show that the preprocessed data one needs to compute an inner product is of size at least proportional to the length of the vectors, while on the other hand the inner product itself is just one field element, so that the cost of doing reconstruction of such a result will be independent of the length of the vectors.

In order to obtain the above result and give more details, we proceed by proving some auxiliary results with lower bounds on the amount of preprocessed data needed for a secure evaluation of a function f.

Lower bounds for secure function evaluation in the preprocessing model. In this section we will give lower bounds for secure implementations of functions $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ from a functionality P_{U_f,V_f} which outputs correlated randomness for the semi-honest model. In particular, we are in the setting where the parties A, B have access to a functionality that gives a random variable U_f to A and V_f to B with some guaranteed joint distribution P_{U_f,V_f} of U_f, V_f. Given this, the parties compute securely a function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ where A holds $x \in \mathcal{X}$, and B holds $y \in \mathcal{Y}$. This function should have no redundant inputs for party A [3] :

$$\forall x, x' \in \mathcal{X}(x \neq x' \to \exists y \in \mathcal{Y} : f(x, y) \neq f(x', y)) \tag{1}$$

The authors of [WW10] obtained Theorem 4 that gives a lower bound on the conditional entropy of P_{U_f,V_f}. Their bound applies for input distributions X and Y which are independent and uniformly distributed. This implies worst case

[3] Party A must enter all the information about X into the protocol. An example of a function that satisfies this property is the inner product IP.

communication complexity. Our bound in Theorem 5 also applies to independent and uniform distributions.

Theorem 4. *Let $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be a function that satisfies property (1). Assume there exists a protocol having access to P_{U_f,V_f} which is an ϵ-secure implementation of f in the semi-honest model with $t = 1$ corruptions. Then*

$$H(U_f|V_f) \geq \max_y H(X|f(X,y)) - (3|\mathcal{Y}| - 2)(\epsilon \log |\mathcal{Z}| + h(\epsilon)) - \epsilon \log |\mathcal{X}| - h(\epsilon).$$

Our general result will only apply to functions where the output lives in a ring $\mathcal{Z}$. As it will become apparent, for the next theorem we require the following property for a function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$:

$$\forall x, x' \in \mathcal{X}(x \neq x' \to \exists y_1, y_2 \in \mathcal{Y} : f(x,y_1) - f(x,y_2) \neq f(x',y_1) - f(x',y_2)) \quad (2)$$

Note that the bound in Theorem 4 still applies for functions f that satisfy properties (1) and (2).

In the following we explore the lower bounds on the amount of preprocessed data with respect to composition of functions. In Theorem 5 we prove a lower bound on the conditional entropy of P_{U_h,V_h} for a function h which is a linear combination of two functions f and g. Our bound also applies to compositions of k functions where k is an arbitrary number. Basically we show that the amount of preprocessed data you need to compute the sum of f and g is the sum of what you need to compute f and g separately, as long as f and g are applied to distinct and independent inputs. We clearly need this assumption, as otherwise the theorem is clearly false, just think of applying $f = g$ on the same inputs.

Theorem 5. *Let $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}_f$, $g : \mathcal{Z} \times \mathcal{W} \to \mathcal{Z}_g$ be functions that satisfy properties (1) and (2). Assume that $\mathcal{Z}_f = \mathcal{Z}_g$. Let h be a linear combination of f and g, namely: $\forall x \in \mathcal{X}, y \in \mathcal{Y}, z \in \mathcal{Z}, w \in \mathcal{W}$, $h(x,z,y,w) := \alpha f(x,y) + \beta g(z,w)$ for some $\alpha, \beta \neq 0$. If there exists a protocol that securely implements the function h with access to P_{U_h,V_h}, then it holds that*

$$H(U_h|V_h) \geq \max_y H(X|f(X,y)) + \max_w H(Z|g(Z,w)) \ .$$

Furthermore, the function h will have the following property:

$$\begin{aligned} &\forall x \neq x' \in \mathcal{X}, z \neq z' \in \mathcal{Z}\ \exists y_1, y_2 \in \mathcal{Y}, w_1, w_2 \in \mathcal{W} : \\ &\qquad h(x,z,y_1,w_1) - h(x,z,y_2,w_2) \neq h(x',z',y_1,w_1) - h(x',z',y_2,w_2) \end{aligned} \quad (3)$$

Proof. We start by proving that the function h has this property:

$$\begin{aligned} &\forall x, x' \in \mathcal{X}, z, z' \in \mathcal{Z}((x,z) \neq (x',z') \to \\ &\qquad \exists y \in \mathcal{Y}, w \in \mathcal{W} : h(x,z,y,w) \neq h(x',z',y,w) \end{aligned} \quad (4)$$

By assumption we consider the following two properties on the function g:

$$\forall z \neq z' \in \mathcal{Z}\ \exists w \in \mathcal{W} : g(z,w) \neq g(z',w) \quad (5)$$

$$\forall z \neq z' \in \mathcal{Z}\ \exists w_1, w_2 \in \mathcal{W} : g(z, w_1) - g(z, w_2) \neq g(z', w_1) - g(z', w_2) \quad (6)$$

and properties (1) and (2).

In order to prove properties (4) and (3) for the function h we proceed as follows:

Case 1. $x = x', z \neq z'$:
Suppose that $\exists y$ such that $f(x', y) = f(x', y)$. By assumption $\exists w \in \mathcal{W}$: $g(z, w) \neq g(z', w)$. Therefore, it follows that $f(x', y) - f(x, y) \neq g(z, w) - g(z', w)$ and property (4) holds.

Case 2. $x \neq x', z = z'$:
Suppose that $\exists w$ such that $g(z', w) = g(z', w)$. By assumption $\exists y \in \mathcal{Y}$: $f(x, y) \neq g(x', y)$. It follows that $f(x', y) - f(x, y) \neq g(z, w) - g(z', w)$ and property (4) holds.

Case 3. $x \neq x', z \neq z'$:
Let $c = f(x', y) - f(x, y)$ for some $y \in \mathcal{Y}$. By assumption $\exists w_1, w_2 \in \mathcal{W}$ such that $c_1 = g(z, w_1) - g(z', w_1)$ and $c_2 = g(z, w_2) - g(z', w_2)$ such that $c_1 \neq c_2$. Without loss of generality, assume that $c \neq c_1$ then $f(x', y) - f(x, y) \neq g(z, w_1) - g(z', w_1)$ and property (3) follows.

Since the function h satisfy property (4) it also has property (1) and hence we get from Theorem 4 that

$$H(U_h|V_h) \geq \max_{y,w} H(X, Z|h(X, Z, y, w)) \ .$$

We then get that:

$$\begin{aligned} H(U_h|V_h) &\geq \max_{y,w} H(X, Z|\alpha f(X, y) + \beta g(Z, w)) && (7)\\ &\geq \max_{y,w} H(X, Z|f(X, y), g(Z, w)) && (8)\\ &\geq \max_{y} H(X|f(X, y)) + \max_{w} H(Z|g(Z, w)) && (9) \end{aligned}$$

Inequality (9) follows from the independence of X, Z. This proves the theorem. □

Remark 1. The above theorem also applies to multiplicative relations ruling out the cases where $g(z, w) = 0$ and $f(x, y) = 0$.

Exploiting Theorem 5 we prove a lower bound for the inner product function IP_k as per Definition 2.

Lemma 1. *Let $\kappa \geq 1$ and let $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be a multiplication function as per Definition 1. If there exist a protocol Π_{IP_k} which securely implements the inner product function IP_k with error probability ϵ in the semi-honest model and having access to $P_{U_{\mathrm{IP}_k} V_{\mathrm{IP}_k}}$ then*

$$H(U_{\mathrm{IP}_k}|V_{\mathrm{IP}_k}) \geq k \cdot \max_{y} H(X|f(X, y)) \quad (10)$$

Proof. Since the function f satisfies properties (1) and (2), a straightforward application of Theorem 5 for $k = 2$ yields $H(U_{\mathrm{IP}_2}|V_{\mathrm{IP}_2}) \geq 2 \cdot \max_y H(X|f(X,y))$. However it is easy to see that the proof of Theorem 5 extends to addition of k functions for any k, so the lemma follows in the same way from this more general result. □

Utilising Theorem 5 in the following we prove that any function whose "preprocessing complexity" is large enough requires lots of communication. What "large enough" means here is determined by the output secret-sharing scheme used in the protocol, in a sense we make precise below. In the following, when f is a function with two inputs and one output, we will speak about *a protocol for computing shares of an f-output*, denoted by $\Pi_{f-output}$. This is essentially the same as a MGP except that we replace multiplication by f. So the protocol takes as input shares of x_1 and x_2 and computes shares of $f(x_1, x_2)$ as output. Note that the inputs x_1, x_2 may be vectors of field elements, whereas we will by default assume that the output is a single field element.

In the sequel, for simplicity of exposition let $\mathtt{L}_f$ denote a lower bound on the amount of preprocessed data needed for a secure implementation of f in the preprocessing model and let $\mathtt{U}_f$ denote an upper bound.

Reconstruction Protocol Π_{rec}. Let $\mathcal{S}_t^n$ be the secret-sharing scheme as per Definition 5 and let $f'_{\mathcal{S}_t^n}$ be the reconstruction function of $\mathcal{S}_t^n$. Then, we can securely implement the function $f'_{\mathcal{S}_t^n}$ in the preprocessing model via the protocol Π_{SPDZ} yielding the protocol Π_{rec}.[4]. It follows that Π_{rec} demands communication and that its complexity depends only on the underlying secret-sharing scheme $\mathcal{S}_t^n$. In this case we obtain an upper bound $\mathtt{U}_{rec}$ on the amount of preprocessed data consumed by Π_{rec}.

Theorem 6. *Consider the preprocessing model where t of the n players may be passively corrupted. Let Π_{rec} be a secure output reconstruction protocol with access to $P_{U_{rec},V_{rec}}$ for the secret-sharing scheme $\hat{\mathcal{S}}_t^n$. Let f be a function with two inputs and one field element as output such that $\mathtt{U}_{rec} < \mathtt{L}_f$. There exists no passively secure n-player protocol $\Pi_{f-output}$ with expected anticipated communication complexity $\leq t$ for computing shares of an f-output with $\hat{\mathcal{S}}_t^n$ as output secret-sharing scheme.*

Proof. We start by assuming a fixed communication pattern. Suppose for contradiction that there exists a protocol Π_f where at most t players communicate. Assume that it is the t first parties. Given two parties A and B, we are going to construct a two-party protocol $\Pi_f^{A,B}$ which on input a, b from A, B, respectively, securely computes $f(a, b)$. The idea is to execute the $\Pi_{f-output}$ protocol in which A emulates the t players who communicate while B emulates the rest of the parties but we are interest just for one additional party, say P_{t+1}. In particular, protocol $\Pi_f^{A,B}(a, b)$ proceeds as follows:

[4] Note that any protocol in the preprocessing model can be used.

Protocol $\Pi_f^{A,B}(a,b)$:

Input Phase

Parties A, B secret share their inputs a, b using the secret-sharing scheme $\mathcal{S}_t^n$. More specifically, A computes $[a]^{\mathcal{S}_t^n} \leftarrow \mathsf{Share}(a,n,t)$ and B computes $[b]^{\mathcal{S}_t^n} \leftarrow \mathsf{Share}(b,n,t)$.

Party A sends the input share $(a_{t+1},\ldots,a_n)$ to party B and Party B sends the input shares $(b_1,\ldots,b_t)$ to party A.

Evaluation Phase

Parties A, B invoke the protocol $\Pi_{f-output}$ where A emulates the t players who communicate, and we assume these are the first t players. This means that this phase involves no communication between A and B, but it may consume some preprocessed data. The execution of $\Pi_{f-output}$ yields a sharing of $[c]^{\mathcal{S}_t^n}$ and outputs $(c_1,...,c_t)$ to party A and $(c_{t+1},\ldots,c_n)$ to party B.

Output Phase

Both parties locally invoke protocol Π_{Rec} with access to $P_{U_{rec},V_{rec}}$ which on input $[c]^{\hat{\mathcal{S}}_t^n}$ outputs the result $f(a,b)$.

Correctness of the protocol follows immediately from the correctness of $\Pi_{f-output}$ and Π_{Rec}. The protocol can be argued to be secure(private) as follows: in the input phase, the parties receive only an unqualified set of shares whose distribution can be simulated perfectly. There is no communication to be simulated in the evaluation phase. In the output phase, simulation is guaranteed by the invocations of the secure protocol Π_{Rec}.

We can claim the following: Note that the communication in $\Pi_f^{A,B}$ is actually independent of the preprocessed data needed in order to securely compute f. Therefore, while reusing the same preprocessed data for each invocation of $\Pi_{f-output}$, we could have executed ℓ instances of $\Pi_f^{A,B}$ on independent inputs without affecting correctness since the simulation is independent of the preprocessed data. However, since protocol Π_{Rec} is interactive its preprocessed data must be refreshed for each of the ℓ executions of Π_{Rec}. This means that the amount of preprocessed data needed in order to compute ℓ instances of f is $\mathtt{U}_f + \ell \cdot \mathtt{U}_{rec}$. So if we consider an ℓ large enough such that $\ell \cdot \mathtt{L}_f > \mathtt{U}_f + \ell \cdot \mathtt{U}_{rec}$, we have a contradiction and the theorem follows.

We now generalize to dynamic communication patterns. As for Theorem 3 we can show that if the expected communication complexity is $\leq t$, then we can find a set of t parties such that with positive probability only these parties communicate. Assume without loss of generality that it is the parties $P_1,\ldots,P_t$. We call the parties $P_{t+1},\ldots,P_n$ the external parties.

As for Theorem 3 we can show that the decision of a party $P_i \in \{P_1,\ldots,P_t\}$ to send a message to an external party or anticipate a message from an external party cannot depend on the private randomness of P_i as the other party of the exchange has to send/anticipate the message at the same time. We can similarly show that the decision cannot depend on P_i's share of a or b, as it would leak information to an external set of t parties: the adversary can corrupt

t external parties to get t shares and then observe whether P_i communicates to get information on the share of P_i, which would leak information on the secret that has been shared. This means that for all $P_i \in \{P_1, \ldots, P_t\}$ there exist a function ϱ_i such that P_i communicates with an external party if and only if $\varrho_i(u_1, \ldots, u_t) = 1$. Similarly there exists a function ϱ_{t+1} such that P_{t+1} communicates at all if and only if $\varrho_{t+1}(u_{t+1}) = 1$.

Let E be the event that $\varrho_i(u_1, \ldots, u_t) = 0$ for $i = 1, \ldots, t$ and that $\varrho_{t+1}(u_{t+1}) = 0$. Let U be the distribution of the shared randomness used by $\Pi_{f-output}$. Since E happens with constant probability and we can determine from the communication patterns whether E occurred, it follows that E occurs with statistically close probabilities in the real execution of $(\Pi_{f-output}, U)$ and in the simulation. Otherwise we could use the presence of E to distinguish the real execution and the simulation. Furthermore, the real execution conditioned on E occurring and the simulation conditioned on E occurring must be statistically close. Otherwise we can distinguish by outputting a random guess when E does not occur and using the distinguisher for the case when E occurs when E actually occurs. It therefore follows that if we look at the distribution of correlated randomness U' which is U restricted to E occurring, then $\Pi_{f-output}$ run with U' is also secure. This protocol can be simulated simply by running the simulator for $(\Pi_{f-output}, U)$ until it produces a transcript where E occurred. The proof then follows as above but using $(\Pi_{f-output}, U')$ instead of $(\Pi_{f-output}, U)$. □

Given a function f with one output and a non-zero lower bound, we can add it to itself on distinct inputs a sufficient number of times in order to satisfy the condition in the above theorem. An example of a function f is the inner product function IP_k which is the composition of k MULT functions. In Lemma 1 we obtained a lower bound $\mathtt{L}_{\mathrm{IP}^k}$ on the amount of preprocessed data consumed by a protocol that securely implements the function IP^k. Now, if k is large enough to satisfy the condition $\mathtt{U}_{rec} < \mathtt{L}_{\mathrm{IP}_k}$, then it holds that $\ell \cdot \mathtt{U}_{rec} + \mathtt{L}_{\mathrm{MULT}} < \ell \cdot \mathtt{L}_{\mathrm{IP}_k}$ for large enough ℓ leading to a contradiction with Theorem 6.

5 Conclusions

We have shown that any protocol that follows the *typical* gate-by-gate design pattern must communicate for every multiplication gate, even if only semi-honest security is required, for both honest majority and dishonest majority with preprocessing where the target secret sharing scheme is additive. We have also shown similar results for any target secret sharing scheme in the dishonest majority setting. This highlights a reason why, even with preprocessing, all know protocols which are efficient in the circuit size $|C|$ of the evaluated function require $\Omega(n|C|)$ communication and $\Omega(d_C)$ rounds where d_C is the depth of C. Our result implies that a fundamental new approach must be found in order to construct protocols with reduced communication complexity and therefore beat the complexities of GMW, SPDZ etc. Of course, it is also possible that our bounds hold for any protocols efficient in the circuit size of the function, and this is the main problem we leave open.

Acknowledgments

Research supported by the Danish National Research Foundation and the National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation and from the Center for Research in Foundations of Electronic Markets (CFEM), supported by the Danish Strategic Research Council. Partially supported by the European Research Commission Starting Grant 279447. This work was done in part while the third author was visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant CNS-1523467.

References

[BFKR90] Donald Beaver, Joan Feigenbaum, Joe Kilian, and Phillip Rogaway. Security with low communication overhead. pages 62–76, 1990.

[BGW88] Michael Or Ben, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). pages 1–10, 1988.

[BI05] Omer Barkol and Yuval Ishai. Secure computation of constant-depth circuits with applications to database search problems. pages 395–411, 2005.

[BSFO12] Eli Ben-Sasson, Serge Fehr, and Rafail Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. pages 663–680, 2012.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). pages 11–19, 1988.

[CK93] Benny Chor and Eyal Kushilevitz. A communication-privacy tradeoff for modular addition. *Inf. Process. Lett.*, 45(4):205–210, 1993.

[DIK+08] Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam Smith. Scalable multiparty computation with nearly optimal work and resilience. pages 241–261, 2008.

[DIK10] Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. pages 445–465, 2010.

[DN07] Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. pages 572–590, 2007.

[DPP14] Deepesh Data, Manoj Prabhakaran, and Vinod M. Prabhakaran. On the communication complexity of secure computation. pages 199–216, 2014.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. pages 643–662, 2012.

[GIP+14] Daniel Genkin, Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and Eran Tromer. Circuits resilient to additive attacks with applications to secure computation. pages 495–504, 2014.

[GIP15] Daniel Genkin, Yuval Ishai, and Antigoni Polychroniadou. Efficient multiparty computation: From passive to active security via secure SIMD circuits. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 721–741, 2015.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. pages 294–304, 2000.

[IKM+13] Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. On the power of correlated randomness in secure computation. pages 600–620, 2013.

[IPS09] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. pages 294–314, 2009.

[NN01] Moni Naor and Kobbi Nissim. Communication preserving protocols for secure function evaluation. pages 590–599, 2001.

[NNOB12] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. pages 681–700, 2012.

[RBO89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). pages 73–85, 1989.

[WW10] Severin Winkler and Jürg Wullschleger. On the efficiency of classical and quantum oblivious transfer reductions. pages 707–723, 2010.