More Communication Lower Bounds for Information-Theoretic MPC

Ivan Bjerre Damgård
Department of Computer Science, Aarhus University, Denmark

Boyang Li
Institute of Interdisciplinary Information Sciences, Tsinghua University, Beijing, China
(This work was done while visiting Aarhus University).

Nikolaj Ignatieff Schwartzbach
Department of Computer Science, Aarhus University, Denmark

Abstract

We prove two classes of lower bounds on the communication complexity of information-theoretically secure multiparty computation. The first lower bound applies to perfect passive secure multiparty computation in the standard model with $n = 2t + 1$ parties of which t are corrupted. We show a lower bound that applies to secure evaluation of any function, assuming that each party can choose to learn or not learn the output. Specifically, we show that there is a function H^* such that for any protocol that evaluates $y_i = b_i \cdot f(x_1, ..., x_n)$ with perfect passive security (where b_i is a private boolean input), the total communication must be at least $\frac{1}{2}\sum_{i=1}^{n} H_f^*(x_i)$ bits of information.

The second lower bound applies to the perfect maliciously secure setting with $n = 3t + 1$ parties. We show that for any n and all large enough S, there exists a reactive functionality F_S taking an S-bit string as input (and with short output) such that any protocol implementing F_S with perfect malicious security must communicate $\Omega(nS)$ bits. Since the functionalities we study can be implemented with linear size circuits, the result can equivalently be stated as follows: for any n and all large enough $g \in \mathbb{N}$ there exists a reactive functionality F_C doing computation specified by a Boolean circuit C with g gates, where any perfectly secure protocol implementing F_C must communicate $\Omega(ng)$ bits. The results easily extends to constructing similar functionalities defined over any fixed finite field. Using known techniques, we also show an upper bound that matches the lower bound up to a constant factor (existing upper bounds are a factor $\lg n$ off for Boolean circuits).

Both results also extend to the case where the threshold t is suboptimal. Namely if $n = kt + s$ the bound is weakened by a factor $O(s)$, which corresponds to known optimizations via packed secret-sharing.

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques

Keywords and phrases Multiparty Computation, Lower bounds

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.2

Funding *Ivan Bjerre Damgård*: The European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 669255 (MPCPRO).
Nikolaj Ignatieff Schwartzbach: The European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 669255 (MPCPRO).

1 Introduction

In secure multiparty computation (MPC) a set of n parties compute an agreed function on inputs held privately by the parties. The goal is that the intended result is the only new information released and is correct, even if t of the parties are corrupted by an adversary.

In this paper we focus on unconditional security where even an unbounded adversary learns nothing he should not, and we ask what is the minimal amount of communication one needs to compute a function securely. To be clear, we will only consider functions where the size of the output is much shorter than the input, so we avoid trivial cases where the communication is large, simply because the parties need to receive a large output. Note that one can always compute the function without security by just sending the inputs to one party and let them compute the function, so the question to consider is: compared to the size of the inputs, what overhead in communication (if any) is required for a secure protocol? Note that a different and probably much harder question is if, in general, the communication must be larger than the circuit size of the function.

These questions only seem interesting for unconditional security: for computational security we can use homomorphic encryption to compute any function securely with only a small overhead over the input size.

There is a lot of prior work on lower bounding the communication required in interactive protocols, and we survey some of this below. However, the most relevant existing work for us is [6] which considers exactly the questions we ask here for the case of honest majority, $n = 2t + 1$, and passive (semi-honest) security. They show that a factor n overhead over the input size is required for a variant of the inner product function, where parties may privately choose to learn or not to learn the output. The result extends to the case of suboptimal threshold where $n = 2t + s$, and then the overhead becomes n/s.

Note that this result leaves open two important questions:

Firstly, a natural next step after the results from [6] is to ask which functions in general require large communication. However, applying the result from [6] to functions other than the inner product is nontrivial because they leverage a particular property of the inner product, namely that it can be used to implement a PIR, which is of course not the case in general. In this work, we therefore ask:

Can we show lower bounds for perfect passive secure evaluation of functions other than the inner product?

Second, it is well known that perfect malicious security can be achieved if and only if $t < n/3$ and the result from [6] has nothing to say about this case: to apply it, one would need to set s to be $\Theta(n)$ and then their lower bound becomes trivial. Hence, the final open question we consider is:

Can we show lower bounds for perfect malicious security in the case where $n = 3t + 1$?

We answer both questions in the affirmative.

1.1 Our results

1.1.1 Bounds for passive security

In this paper, we prove lower bounds for the model with n parties of which t are statically corrupted. The network is synchronous, and we assume that the adversary can learn the length of any message sent (in accordance with the standard ideal functionality modeling secure channels which always leaks the message length). We consider information-theoretically secure protocols with static corruption in the maximal threshold model.

On the technical side, what we show are actually lower bounds on the entropy of the messages sent on the network when the inputs have certain distributions. This then implies similar bounds in general on the average number of bits to send: an adversary who corrupts no one still learns the lengths of messages, and must not be able to distinguish between different distributions of inputs. Hence message lengths cannot change significantly when we change the inputs, otherwise the protocol is insecure.

For our passive lower bounds, we require that protocols securely implement the standard functionality for secure function evaluation, where we add the option that each player P_i can privately choose to learn or not to learn the output, by selecting an additional input bit b_i. What we show is that there exists a mapping H^* which takes any function f, such that in any n-party protocol securely evaluating the output $y_i = b_i \cdot f(x_1, \cdots, x_n)$ for player P_i, the total average communication must be at least $\frac{1}{2}\sum_{i=1}^{n} H_f^*(x_i)$ where x_i is the input of player P_i.

Very roughly speaking, the function $H_f^*(x_i)$ measures how much uncertainty remains in the function output given that we know x_i. Specifically, it is defined as the maximum uncertainty that remains on any subset of inputs of size t among the remaining $2t$ inputs. The lower bounds that we establish are tight in some cases: for the inner product we get a bound of $\Omega(n)$ times the input length, so we recover the lower bound of [6]. Since the inner product can be computed by a circuit of linear size, this bound is tight up to constant factors. For the XOR function we get a trivial lower bound which only states that each party must communicate their input. As the XOR function is linear, it is of course not surprising that the bound is trivial in this case. Indeed, for two parties the bound is tight since a passively secure protocol is for one of the two parties to simply reveal their input to the other party. A final interesting example is a function is called "ranking", that provides each party with the index of their input in the sorted list of all inputs. For this example, we get again a non-trivial bound of $\Omega(n)$ times the input size. This bound may not be tight, assuming there is no linear-sized circuit for sorting integers.

On the technical side, our bound is established by considering a fixed party P_i and choosing a bipartitioning of the remaining $2t$ parties into two groups of size t. We show that the entropy such a group provides to the function output is a lower bound on the communication of party P_i so we choose the maximum value among all such partitions. This corresponds to the definition of the function H_f^* mentioned above. Since the adversary is not allowed to distinguish between different distributions of messages we can essentially add all the lower bounds for the communication of all parties to obtain our lower bound.

The lower bound extends to the case where the threshold is submaximal, i.e. $n = 2t + s$. The bound can be established by considering a partitioning of the parties into sets of size s (it is allowed that a party belongs to no set). For each such set, we take the supremum over all ways of bipartitioning the remaining $2t$ parties into two sets of size t to get a communication bound. Since the adversary is not allowed to distinguish between different distributions of messages, again we can add the communication for each such set of size s. This means we get a communication lower bound for each such partition of the parties into sets of size s so we take the maximum among all such partitions. For functions which are "symmetric", any partition of the parties into sets of size s gives the same lower bound which means the final supremum can be omitted. In this case, the lower bound is weakened by a factor $O(s)$ such that the total communication can be shown to be $\Omega(\sum_{i=1}^{n} H_f^*(X_i))/s$. For functions which are not symmetric it is not possible to make a general statement about what happens in the submaximal threshold case, though highly asymmetric functions likely have weaker lower bounds since only a few parties contribute a large amount of entropy to the function output.

1.1.2 Bounds for active security

For our lower bounds for active security, we make use of the Universal Composability (UC) model for secure protocols. Recall that, in the UC model, we specify an "ideal functionality" in order to state what a protocol is supposed to do. The functionality accepts input from the parties and computes outputs in a specific way that an adversary by definition cannot modify. A protocol securely implements the functionality if running the protocol is, in a certain well-defined sense, "equivalent" to interacting with the functionality.

In our case, we consider a functionality F_f computing a function f with a specific structure. Namely, the functionality first receives input from all parties and sends an acknowledgement to everyone. Then it receives a second batch of inputs, computes the desired function and sends the result to all parties. This structure of F_f implies that in any protocol implementing F_f, the first set of inputs must be chosen and committed before the second set of inputs are chosen, and this is important for the proof of our lower bound. However, even if this particular structure is a limitation, the model still covers some natural applications. For instance, the concrete function we study models a case where a long string (a database) is determined in the first phase, and the function to be evaluated then returns a bit in a certain position chosen later (an entry in the database).

We assume UC security mainly for simplicity of exposition, we can actually make do with significantly weaker assumptions, this is detailed in Section 3.5. What we show is that for all n and any sufficiently large S, there exists a function f_S with input size S such that any protocol that evaluates F_{f_S} securely must communicate $\Omega(nS)$ bits.

Even more is true: we are able to construct functions f_S as we just claimed such that they can be evaluated by circuits of size $O(S)$. This means we also get the following result: for any n and all large enough $g \in \mathbb{N}$ there exists a Boolean circuit C with g gates specifying the computation to be done by functionality F_C, such that any protocol that evaluates F_C securely must communicate $\Omega(ng)$ bits.

We emphasize that our result leaves open the question of overhead over the circuit size when the circuit is much bigger than the inputs. However, there is still something we can say about this general question. Note that the general MPC protocols we know are not, strictly speaking, protocols. Rather, they are protocol compilers that take a circuit C as input, and produce a protocol for computing C securely. Our results do imply that any such compiler must produce a protocol with large communication overhead over the circuit size when applied to circuits in the family we build. Now, if this overhead would no longer be present when applying the compiler to other circuits, it would mean that it was able to exploit in some non-trivial way the structure of the circuit it is given. Doing this would require protocol compilers of a completely different nature than the ones we know, which do "the same thing" to any circuit they are given.

This bound also extends to the case where the threshold t is suboptimal. Namely, if $n = 3t + s$, then the lower bound is $O(ng/s)$ and this shows that the improvement in communication that we know we can get using so-called packed secret sharing, is the best we can achieve. The bound does not, however, extend to statistical security. We show in Section 3.6 that there exists a statistically secure protocol breaking the bound already in the 4-party case.

We also show an upper bound that matches the lower bound up to a constant factor for all values of $t < n/3$. This is motivated by the fact that the existing upper bound from [14] is a factor $\lg n$ off for Boolean circuits. We do this by exploiting recent results by Cascudo et al. [4] on so-called reverse multiplication friendly embeddings. Other than establishing the exact communication complexity for this particular class of functions, it also shows that our result is the best possible general lower bound we can have.

To show our results, we start from a lower bound for the communication complexity of a specific function for the case of four parties including one maliciously corrupt player. We then "lift" this result to the multiparty case. This high-level strategy is similar the one used in [6], however our proof for the four party case as well as the concrete lifting technique are very different from what was done in [6]. In fact it is easy to see that new techniques are necessary to achieve our result. Namely, in our case where $t < n/3$, [6] only gives a trivial result, as mentioned above. Nevertheless [6] is known to be optimal for passive security, even in the case of suboptimal threshold. This means that there is no way to use their proof for our question, one must somehow exploit the fact that the considered protocols are assumed to be maliciously secure.

1.2 Related work

Prior work on lower bounding communication in interactive protocols includes [15, 12, 5, 11, 15, 16, 2, 13] (and see [10] for an overview of these results). The previous work most relevant to us is [10]. They consider a special model with three parties where only two have input and only the third party gets output, and consider perfect secure protocols. This paper was the first to show an explicit example of a function where the communication for a (passive and perfectly) secure protocol must be larger than the input.

Later, in [8], a lower bound was shown on the *number of messages* that must be sent to compute a certain class of functions with statistical security. When the corruption threshold t is $\Theta(n)$, their bound is $\Omega(n^2)$. This of course implies that $\Omega(n^2)$ bits must be sent. However, we are interested in how the communication complexity relates to the input and circuit size of the function, so once the input size becomes larger than n^2 the bound from [8] is not interesting in our context.

In [9], lower bounds on communication were shown that grow with the circuit size. However, these bounds only hold for a particular class of protocols known as gate-by-gate protocols, and we are interested in lower bounds with no restrictions on the protocol.

2 Lower bounds for arbitrary functions

In this section we prove a lower bound on the communication complexity for perfect passive secure multiparty computation. The lower bound applies to any function in which the parties can choose to learn or not to learn the output. For some functions, the lower bound can be shown to be tight.

Let X be a random variable with pdf $p : \mathcal{X} \to [0, 1]$. We define the (Shannon) entropy of X as:

$$H(X) = -\sum_{x\in\mathcal{X}} p(x)\ \lg p(x)$$

where lg is base 2. The entropy measures the uncertainty of X: to communicate the outcome of X, an average of $H(X)$ bits have to be communicated. We define the conditional entropy $H(Y \mid X)$ as the amount of information in Y left, given that we know X. If $H(Y;X)$ is the joint entropy we define:

$$H(Y \mid X) = H(Y;X) - H(X)$$

A related measure is the *mutual information* $I(Y;X)$ that measures how much information the two random variables X, Y have in common. It is defined as:

$$I(X;Y) = H(X) - H(X \mid Y)$$

We will use this measure for our lower bound, we need the following identities:

▶ **Lemma 1.** *$I(X;Y) = I(X;Z) + [H(X \mid Z) - H(X \mid Y)]$.*

Proof. Follows from the definition of mutual information:

$$I(X;Y) = H(X) - H(X \mid Y) = I(X;Z) + H(X \mid Z) - H(X \mid Y) \qquad \blacktriangleleft$$

▶ **Lemma 2.** *Let X, Y, Z be random variables such that $I(Y;Z) \geq H(Y)$. Then $H(X \mid Z) \leq H(X \mid Y)$.*

Proof. By using the chain rule for entropy twice we get:

$$\begin{aligned} H(X,Y,Z) &= H(Y) + H(X \mid Y) + H(Z \mid Y, X) \\ &= H(Z) + H(X \mid Z) + H(Y \mid Z, X) \end{aligned}$$

Since $I(Y;Z) \geq H(Y)$ we have $H(Y \mid Z, X) = 0$ and so we find that:

$$H(X \mid Y) = H(X \mid Z) + [H(Z) - H(Z \mid Y, X)] - H(Y)$$

Noting that $H(Z \mid Y, X) \leq H(Z \mid Y)$ we get:

$$H(Z) - H(Z \mid X, Y) \geq H(Z) - H(Z \mid Y) = I(Y;Z) \geq H(Y)$$

In particular, we have $[H(Z) - H(Z \mid Y, X)] - H(Y) \geq 0$ which concludes the proof. ◀

▶ **Lemma 3.** *Let X, Y, Z be random variables such that $I(X;(Y,Z)) \geq \ell$, and $I(X;Y) = 0$. Then $H(Z) \geq \ell$.*

Proof. We use the chain rule for mutual information to obtain:

$$\ell \leq I(Y, Z; X) = I(X;Y) + I(Z;X \mid Y)$$

By assumption we have $I(X;Y) = 0$. The bound $I(Z;X \mid Y) \leq H(Z)$ is not hard to see and concludes the proof. ◀

We now define the *functional entropy* of a random variable which is used to establish our communication lower bound. Informally, the functional entropy measures how much uncertainty an input to a function provides to its output.

2.1 Functional entropy

We start by considering the binary case: let $f : \mathcal{X} \times \mathcal{X} \to \mathcal{Y} \times \mathcal{Y}$ be a binary function, and let X_1, X_2 be random variables over $\mathcal{X}$. We define the f-expansion of X_1 to be the following exponential-sized string (the case for X_2 is similar):

$$\mathcal{E}_f(X_1) = \mathop{\Big\|}_{x_2 \in \mathcal{X}} f(X_1, x_2)$$

where $\|$ denotes string concatenation. The order of the concatenation matters in the sense that it must be a fixed order, but the specific order is not important. We now define the *functional entropy* of X_i as:

$$H_f(X_i) = H(\mathcal{E}_f(X_i))$$

Loosely speaking, this quantity measures how much uncertainty remains in the function output, given that we remove all randomness from variables other than X_i. Since the value of $\mathcal{E}_f(X_i)$ can be computed from X_i, the functional entropy must be upper bounded by the regular Shannon entropy, i.e. $H_f(X_i) \leq H(X_i)$.

We now extend the notion to an n-ary function $f : \mathcal{X}^n \to \mathcal{Y}^n$ for $n = 2t + 1$ and some t. Let $T \subset \{1, 2, \ldots n\}$ be a set of indices, and define $-T$ as its complement. Note that we can write any f as f' where

$$f(X_1, X_2, \ldots X_n) = f'(X_T, X_{-T})$$

We define the functional entropy of a set of random variables T as:

$$H_T = H_{f'}(X_T)$$

Finally, we define the maximum functional entropy of a variable X_i as:

$$H^*_f(X_i) = \max_{T,\, |T|=t,\, i \notin T} H_T$$

Loosely speaking, $H^*_f(X_i)$ measures how much uncertainty we can have in the function output, if we fix all but t inputs, where these t inputs do not include X_i. We will use this quantity to establish our lower bound.

2.2 Lower bound, arbitrary functions, maximal threshold

In this section we establish the communication lower bound for perfect security and maximal threshold. Let $n = 2t+1$ be an integer, and consider a set of parties $P_1, P_2, \ldots P_n$ computing a function where the i^{th} party learns $y_i = b_i \cdot f_i(\mathbf{X})$, where $b_i \in \{0, 1\}$ is a private boolean input, and $f : \mathcal{X}^n \to \mathcal{Y}^n$ is a vector function. Consider a partition of the n parties into groups of size $t, t, 1$ where X_1 is the concatenated inputs of parties in the first group, X_2 the second group, and X_3 is the input of the single party (not including the b_i inputs). Let $C_{i,j}$ be the (ordered) concatenation of all messages sent between groups i and j.

In the following two lemmas we consider a situation where only the single players in group 3 learns the output, while all other players have their output selection bit set to 0. Now, since no group has more than t players, the adversary can corrupt all players in each single group, and hence neither the first, nor the second group must learn anything new from the protocol.

▶ **Lemma 4.** $I(X_1; (C_{1,2}; C_{1,3})) \geq H_f(X_1)$.

Proof. By privacy against group 1, the variables $C_{1,2}, C_{1,3}$ are independent of X_2 and X_3. This means group 2 and P_3 can resample randomness and use $(C_{1,2}, C_{1,3})$ as an oracle to compute $f(x_1, x_2, x_3)_{2,3}$ for any values of x_2, x_3. This implies that the mutual information between the communication and the expansion is at least the entropy of the expansion:

$$I(\mathcal{E}_f(X_1); (C_{1,2}; C_{1,3})) \geq H(\mathcal{E}_f(X_1)) = H_f(X_1) \tag{1}$$

On the other hand, as $\mathcal{E}_f(X_1)$ is determined by X_1 we have that,

$$I(X_1; \mathcal{E}_f(X_1)) = H_f(X_1) \tag{2}$$

We now compute:

$$\begin{aligned} & I(X_1; (C_{1,2}, C_{1,3})) \\ & = I(X_1; \mathcal{E}_f(X_1)) + (H(X_1 \mid \mathcal{E}_f(X_1)) - H(X_1 \mid (C_{1,2}; C_{1,3})) && \text{by Lemma 1} \\ & = H_f(X_1) + [H(X_1 \mid \mathcal{E}_f(X_1)) - H(X_1 \mid (C_{1,2}; C_{1,3}))] && \text{by Equation (2)} \end{aligned}$$

By Equation (1) we can apply Lemma 2 to conclude the value in the brackets is nonnegative which concludes the proof. ◀

▶ **Lemma 5.** $H(C_{1,3}) \geq H_f(X_1)$.

Proof. Immediate consequence of Lemmas 3 and 4 because of privacy against group 2 which implies $I(X_1; C_{1,2}) = 0$. ◀

▶ **Theorem 6.** *In any MPC protocol of maximal threshold $n = 2t + 1$ that evaluates $y_i = b_i \cdot f_i(x_1, x_2, \ldots, x_n)$ with perfect passive security, the total communication is at least*

$$\frac{1}{2}\sum_{i=1}^{n} H_f^*(X_i)$$

bits of information.

Proof. Consider any party P_i. Then for any partition of the remaining $2t$ parties into two groups of size t, Lemma 5 gives a lower bound on $H(C_{1,3})$ for a certain setting of the inputs. We then choose the maximum such lower bound which is precisely $H_f^*(X_i)$. By perfect passive security, the adversary is not allowed to distinguish between different distributions of messages so we can add the lower bound obtained for each choice of P_i. Finally, we divide by two because each bit is counted exactly twice: once at the sender and once at the receiver. ◀

2.3 Lower bound, arbitrary functions, submaximal threshold

In this section we consider the case when the number of corruptions is submaximal, i.e. $n = 2t + s$ for some $s > 1$. We extend our definition of H_f^* to apply to groups of variables. Let S be some group of parties of size s we then define:

$$H_f^*(X_S) = \max_{T,\, |T|=t,\, S\cap T=\emptyset} H_T$$

We consider a fixed partition of the parties into groups of size t, t, s: we call the concatenated inputs of the parties in each group for X_1, X_2, X_3, and let $C_{i,j}$ denote the correspondence between groups i, j.

Let $\mathcal{S}$ be a partition of the parties into sets of size s, and let $\mathscr{S}$ be the set of all such partitions. Note that a party is allowed to belong to no set in $\mathcal{S}$, say if $2t$ is not divisible by s.

▶ **Theorem 7.** *In any MPC protocol of submaximal threshold $2t + s$ that evaluates $y_i = b_i \cdot f_i(x_1, x_2, \ldots, x_n)$ with perfect passive security, the total communication is at least*

$$\max_{\mathcal{S}\in\mathscr{S}} \left[\frac{1}{2}\sum_{S\in\mathcal{S}} H_f^*(X_S)\right]$$

bits of information.

Proof. Consider some fixed partition $\mathcal{S}$ of the parties into sets of size s. We can let any element $S \in \mathcal{S}$ define a partition of the parties into sets of size t, t, s. The third group can be regarded as a single party with the concatenated inputs as their input. In doing so, we obtain the result of Lemma 5 for any such partition. This means the communication for the third group must be at least $H_f^*(X_S)$. Since the adversary is not allowed to distinguish between different distributions of messages, we can add the communication for all $S \in \mathcal{S}$ to get a lower bound on the communication. Finally, any such $\mathcal{S}$ yields a lower bound, so we choose the partition $\mathcal{S} \in \mathscr{S}$ that maximizes the lower bound. ◀

The statement of the theorem allows for the function to be "asymmetric" in the sense that some ways of partitioning the parties gives stronger bounds. If any choice of $\mathcal{S}$ gives the same lower bound, we say the function f is symmetric. For symmetric functions, the above lower bound can be simplified. We assume that s divides n for simplicity.

▶ **Corollary 8.** *Let f be a symmetric function. Then in any MPC protocol of submaximal threshold $2t+s$ that evaluates $y_i = b_i \cdot f_i(x_1, x_2, \dots, x_n)$ with perfect passive security, the total communication is at least*

$$\frac{n}{2s} H_f^*(X_1)$$

bits of information.

2.4 Examples

We briefly provide some examples of different choices of f. In the following, we let I denote an upper bound on the bit length of each participants input, i.e. $X_i \in \{0,1\}^I$ for every i.

2.4.1 Inner product

We consider a variant of the inner product function where $2t$ parties provide inputs, while the last party only provides a value for b. Now consider any single party. We can then divide the remaining $2t$ parties along the "aisle" of the inner product function. Closer study reveals that almost all information in X_i matters, meaning we get:

$$H_f^*(X_i) = tI$$

Summing this up reestablishes the lower bound of [6]. We note that this lower bound is tight up to constant factors.

2.4.2 XOR

Consider the bitwise XOR function, that takes n inputs $x_1, \dots x_n \in \{0,1\}^I$ and outputs $y = \bigoplus_{i=1}^n x_i$. We note that two inputs X_T, X'_T provide the same expansion iff $\bigoplus_{i \in T} X_i = \bigoplus_{i \in T} X'_i$. This means we get:

$$H_f^*(X_i) = I$$

Summing this up gives a lower bound of $\Omega(nI)$ which only states that each party must communicate their input. However, for two parties this is tight since a passively secure two-party protocol for XOR is for one of the parties to simply reveal their input.

2.4.3 Ranking

Consider a function where each party P_i inputs an integer x_i and learns the index of their input in the sorted list of all inputs. Note that two inputs X_T, X'_T have the same expansion if and only if they are permutations of each other. Strictly speaking, the "if" part is not necessary to be proven since we are calculating a lower bound (not the upper bound), but to illustrate, from perspective of outside T, the output mostly does not change when values within T are permuted among themselves. (This could change some tie-breaking if the mechanics depend on index, but it is true if we allow ties). For the "only if" part, for any two X_T and X'_T that are different in the multi-set of values they contain, there must be an integer that is larger than a values in the multi-set of $\{X_T\}$ and b for $\{X'_T\}$ and $a \neq b$: thus there exists an entry in the expansion where someone outsides T holds the integer, and it would rank differently the two cases, thus the expansion is different. For a list of n items, the information content of a permutation on n elements is bounded by $\lg n! \leq \lg n^n \leq n \lg n$. This means we get a communication lower bound of $\Omega(ntI - nt\lg t)$ bits, which for large inputs is $\Omega(ntI)$.

Regarding the corresponding upper bound, we can use a construction by Parberry ([17]) of a sorting network with $O(n\,(\lg n)^2)$ gates. We can now use any passively secure MPC protocol with linear complexity per gate to compute ranking in time $O(ntI\,(\lg n)^2)$. This has a discrepancy of a factor $O(\lg n)^2$ and gives a communication lower bound of $\Omega(n/(\lg n)^2)$ per gate. This bound is not tight unless there is a passively secure MPC protocol for sorting with sublinear communication complexity per gate; or if there is a circuit for sorting with linear size. The latter is not true unless sorting can be done in linear time. As a result, it is unlikely that our bound is tight for the ranking function.

3 Lower bounds for malicious security

In this section we prove that there is an n-party functionality that can be described by a circuit with g gates such that each party needs to communicate at least $\Omega(g)$ bits. We show this using a series of lemmas that bound the entropy on the communication. We first show the special case for four parties, and then "lift" this to the general case with n parties.

Let $\mathsf{P}_1, \dots \mathsf{P}_n$ be parties connected by pairwise secure channels. We denote by I the *input size* (in bits) of each party, and O the *output size*. For simplicity we assume all parties receive the same output, and denote by $f : \{0,1\}^{nI} \to \{0,1\}^O$ the function to compute.

We assume an active adversary that is allowed to statically corrupt up to t parties where $3t < n$. To define security we use the universal composability (UC) model by Canetti. A quick reminder (for details, see [3]): The model includes the environment Z, a machine that models everything that is external to the protocol, include adversarial attacks. π stands for the protocol, i.e., a set of machines modelling the parties that executes it. The symbol $\diamond$ stands for "compose", so $Z \diamond \pi$ denotes the "real process" where Z interacts with (attacks) the protocol. The model also includes an ideal functionality F that specifies what the protocol is supposed to do. To compare F to π, we need a simulator S that in a nutshell converts the interface offered by F to the interface Z sees when attacking the protocol. Thus $Z \diamond S \diamond F$ denotes the "ideal process" where Z interacts with S and F and hence only attacks allowed by F are possible. We then say that π securely implement F, if there exists a simulator S so that no environment Z can tell if it is doing the real or the ideal process. A bit more formally:

▶ **Definition 9** (UC Security). *A protocol π is said to securely realize a functionality F with perfect malicious security if there exists a simulator S such that for any environment Z, we have that $Z \diamond \pi$ is perfectly indistinguishable from $Z \diamond S \diamond \mathsf{F}$.*

We will consider protocols that implement a reactive ideal functionality F_f for computing f securely. The functionality first receives input from each party, and sends an acknowledgement to all parties once the inputs have been received. Finally, it accepts an additional input from all parties, it then computes the function and sends the output to all parties. As we shall see, it is important towards proving our lower bound that we consider this reactive case, rather than the simpler version where the functionality gets all the inputs in one go.

Note that any protocol implementing F_f will naturally consist of two phases: one that implements the part where the first inputs are sent, which we call the input phase, and the rest, which we call the computation phase. This implies that the first batch of inputs are committed in the first phase, before any information on the second batch of inputs or the output is revealed.

Note that the structure imposed by our choice of F_f still allows us to model quite natural tasks. The concrete function we consider below is one where a long bit string (a "database") is committed in the first phase, and then the function computed will securely extract a particular entry in the database.

3.1 Lower bound, malicious security, four parties

We start by considering a special case of active MPC with four parties $\mathsf{P}_1, \dots \mathsf{P}_4$. In the input phase, the functionality receives an input bit string X_i from each P_i. We assume that $X_1 \in \{0,1\}^I$ (we do not need to assume anything about the lengths of the other inputs). Let L be the length of the concatenation $\mathbf{X} = X_1||X_2||X_3||X_4$. In the second phase, the functionality receives an integer u_i from P_i, where $u_i \in Z_L$. It outputs $(u, \mathbf{X}[u])$, where $u = \sum_i u_i \bmod L$.

We call this function $f_{I,4}$. It has the important property that if the input X_1 of P_1 is changed, there is always a setting of the other inputs for which the change of X_1 will cause the output to change, namely if u points to a position in X_1 that was changed. One consequence of this is the following lemma:

▶ **Lemma 10.** *Assume protocol π computes n-party function f with perfect security, and it is the case that for any $x_1' \neq x_1$, there are values $x_2, ..., x_n$ of the other inputs such that $f(x_1, ..., x_n) \neq f(x_1', x_2, ..., x_n)$. Assume further that P_1 has input x_1, is corrupt but plays honestly. Then the simulator for π must always send x_1 as input to the functionality for f on behalf of P_1.*

Proof. If all players are honest and have inputs $x_1, ..., x_n$, then by perfect security the output must be $f(x_1, ..., x_n)$. If instead P_1 is corrupt but plays honestly, the protocol does exactly the same as if all players are honest so the output is still $f(x_1, ..., x_n)$. Hence, when simulating this case, the simulator must send x_1 to the functionality, for any other value x_1' it may send, the output in the simulation will be incorrect for some choice of $x_2, ..., x_n$, by assumption in the lemma. ◀

Before continuing, we define some terminology: suppose we are given a player P that takes part in a protocol π, and let t be a transcript, that is, the ordered set of all messages sent and received during an execution of the protocol. Now, *sampling random coins consistent with* t means to sample uniformly a random tape r that could have been used to create t if P had done the protocol honestly. In other words, r has the property that if P starts π with random tape r and receives in each round the messages specified in t, he would send the messages specified in t in each round. Of course, such a sampling is not always efficient, but remember that we consider perfectly secure protocols that must be robust against unbounded adversaries.

▶ **Theorem 11.** *In any reactive protocol that implements $\mathsf{F}_{f_{I,4}}$ with perfect malicious security, P_4 must use average communication $\Omega(I)$.*

Proof. Consider a protocol π that computes the function with perfect security. We will consider the messages sent in π as random variables as follows: fix the inputs of $\mathsf{P}_2, \mathsf{P}_3$ and P_4 to arbitrary values x_2, x_3, x_4, and let the input of P_1 be chosen uniformly. Assume π is executed such that all parties follow the protocol. Now, we let T_i for $i = 1, 2, 3, 4$ be the random variable that represents concatenation of all messages sent to and from P_i in the execution of the input phase.

Since the communication pattern must not depend on the inputs, it suffices to show that $\mathrm{H}(T_4) \geq \mathrm{H}(X_1)$. We first show this follows from the following two equations:

$$\mathrm{H}(X_1 \mid T_2) = \mathrm{H}(X_1) \tag{3}$$

$$\mathrm{H}(X_1 \mid T_2, T_4) = 0 \tag{4}$$

To see this, we apply the chain rule for Shannon entropy:

$$\mathrm{H}(T_4) \geq \mathrm{H}(T_4 \mid T_2) + \mathrm{H}(X_1 \mid T_2, T_4) = \mathrm{H}(X_1, T_4 \mid T_2) \geq \mathrm{H}(X_1 \mid T_2) = \mathrm{H}(X_1)$$

We now show each claim separately:

1. Perfect malicious security implies there is a simulator for a corrupt P_2 that plays honestly. The messages created by the simulation are distributed exactly as in a real execution. However, while simulating the input phase, the simulator does not have access to the output, and hence has no information on X_1. It follows that $\mathrm{H}(X_1 \mid T_2) = H(X_1)$.
2. Suppose for the sake of contradiction that X_1 is not determined by T_2, T_4. This means there must exist (at least) two different executions of the input phase where P_1 has different inputs, but the messages seen by $\mathsf{P}_2, \mathsf{P}_4$ are the same. More formally, there exist sets of values of (T_1, T_2, T_3, T_4), say (t_1, t_2, t_3, t_4) and (t'_1, t_2, t'_3, t_4) both with non-zero probability where the first case can occur with $X_1 = x_1$ and the second with $X_1 = x'_1$, where $x_1 \neq x'_1$. We define a value e such that $x_1[e] \neq x'_1[e]$. Now consider the following two attacks on the input phase, represented by environments Z, Z':
   a. Z chooses inputs x_1, x_2, x_3, x_4 for the respective parties, corrupts P_3, but lets her plays honestly in the input phase. If at the end of the input phase P_3 obtains transcript $T_3 = t_3$, she will pretend that she saw $T_3 = t'_3$ instead. She samples random coins r'_3 consistent with t'_3 and completes the protocol honestly, assuming that her view of the input phase was (x_3, r'_3, t'_3). In the last phase, Z sets the inputs u_i in some fixed way such that $e = \sum_i u_i \bmod L$.
   b. Z' chooses inputs x'_1, x_2, x_3, x_4 for the respective parties, corrupts P_1, but lets her plays honestly in the input phase. If at the end of the input phase P_1 obtains transcript $T_1 = t'_1$, she will pretend that she had x_1 as input and saw $T_1 = t_1$ instead. She samples random coins r_1 consistent with t_1 and completes the protocol honestly assuming her view of the input phase was (x_1, r_1, t_1). In the last phase, Z sets the inputs u_i in some fixed way such that $e = \sum_i u_i \bmod L$.

   We can now observe that when the real protocol executes in the first attack, with non-zero probability, it is the case that P_1 has input x_1 and transcripts t_1, t_2, t_3 and t_4 were produced in the input phase. Likewise in the second attack it may happen that P_1 received input x'_1 and transcripts t'_1, t_2, t'_3 and t_4 were produced in the input phase.

   Assuming these events, we see that the protocol execution after the input phase will be the same in the two scenarios: in both cases the players will do the last part of the protocol honestly starting from views (x_1, r_1, t_1), (x_2, r_2, t_2), (x_3, r'_3, t'_3), (x_4, r_4, t_4), where all random coins are uniform, given the corresponding transcripts. Since these views are identically distributed in the two cases and the inputs chosen in the last phase are the same, the same output distribution D is generated in both cases.

   Now consider the simulation of the two attacks. Note that the ideal functionality always computes the output from x_1, x_2, x_3, x_4 in the first case, and from x'_1, x_2, x_3, x_4 in the

second, by Lemma 10. This means that the output is $x_1[e]$ in the first case and $x'_1[e]$ in the second. Assume without loss of generality that $x_1[e] = 0$ and $x'_1[e] = 1$.
On the other hand, we have just seen that the real protocol may sometimes generate output distribution D under both the first and the second attack. Clearly, the probability that D outputs 0 is non-zero, or the probability of output 1 is non-zero. Assume the second case, without loss of generality.
Now, Z can break perfect security: if it sees output 0, it guesses that it has been talking to the simulation, and otherwise it guesses that it is in the real case. Clearly Z will always guess simulation in the ideal (simulated) case but will guess real with non-zero probability in the real case, contradicting perfect indistinguishability. ◀

▶ Remark 12. We can now explain why it is not clear that our proof technique would work if we had used the standard functionality for secure function evaluation where all inputs are given in one go: In order to show that the input phase can produce the same state for the protocol from both input x_1 and x'_1 for P_1, we need to restrict to a particular subset of the transcripts that might occur. But if that same phase also includes provision of the inputs u_i and perhaps the computation of u, the possible values of u might be similarly restricted, so it is not clear that the environment can still choose the index e so that it will "catch" the difference between x_1 and x'_1.

3.2 Lower bound, malicious security, n parties, maximal threshold

We now show that the bound generalizes to multiple parties. Let $n = 3t + 1$ and denote the parties by $\mathsf{P}_{1,1}, \ldots \mathsf{P}_{1,t}, \mathsf{P}_{2,1}, \ldots \mathsf{P}_{2,t}, \mathsf{P}_{3,1}, \ldots \mathsf{P}_{3,t}, \mathsf{P}_4$. Define by $\mathrm{IP}_{I,n}$ the following functionality: each party first provides an I-bit input. When all inputs have been received they are concatenated to form $\mathbf{X}$, then each party provides number $u_i \in Z_L$ where L is the length of $\mathbf{X}$. We set $u = \sum_i u_i \bmod L$ and $(u, \mathbf{X}_u)$ is returned.

▶ **Lemma 13.** $\mathrm{IP}_{I,n}$ *can be computed by a circuit C with $O(nI)$ gates.*

Proof. Let $S = nI$ be input size and assume for simplicity that $S = 2^k$ is an exact power of two. We assume the circuit takes $O(\lg S)$ additional bits which we will denote by $\mathbf{r}$, it corresponds to the index u above. Strictly speaking, we should take the u_i as input and compute their sum modulo L, but the size of the circuit for doing this is insignificant, it is clearly $o(nI)$ for all large enough I. We now proceed using induction in k:

- Base-case $k = 0$: the circuit simply outputs its input bit. This is clearly uniform in the input.
- Induction $k > 0$: we may split the input into two 2^{k-1} sized halves $\mathbf{X}_0$, and $\mathbf{X}_1$. By induction there are circuits C_0, C_1 each with $O(2^{k-1})$ gates computing $\mathbf{X}_0, \mathbf{X}_1$, let y_0, y_1 be the output gates. It suffices to combine C_0, C_1 using a constant number of gates. We now construct the circuit $y = (y_0 \wedge \overline{\mathbf{r}_k}) \vee (y_1 \wedge \mathbf{r}_k)$: this takes at most four gates which is clearly constant. In addition both C_0, C_1 choose their elements uniformly at random: if $\mathbf{r}_k$ is indeed a random bit then y is also uniform.

The result now follows since $t = \Theta(n)$. ◀

▶ **Lemma 14.** *Any reactive protocol that realizes* $\mathrm{IP}_{I,n}$ *with perfect malicious security must have total average communication* $\Omega(ntI)$.

Proof. Consider any party P. We may group the remaining $3t$ parties arbitrarily into 3 groups, each consisting of t parties to produce a functionality equivalent to $F_{f_{tI,4}}$ where P plays the role of P_4. Corrupting any party in the 4-party case corrupts at most t parties in

$\mathrm{IP}_{I,n}$, and the inputs of a t party group are formed by combining the inputs of the individual players in the group (using concatenation or addition modulo L). By this translation, the player P_1 is the 4-player setting has an input of length tI in the first phase, and hence, by Theorem 11, P must communicate at least $\Omega(tI)$ bits. We can apply this argument to each of the $3t+1$ parties and add their resulting communications. It should be noted that this counts every bit *twice*: once at the sender, and once at the receiver, however this has no effect on the asymptotic complexity. We conclude the total average communication is $\Omega(ntI)$ bits. ◀

▶ **Theorem 15.** *There is a (familiy of) Boolean circuit(s) C with g gates such that any reactive n-party protocol computes C with perfect malicious security must use total communication $\Omega(ng)$.*

Proof. Follows immediately from Lemmas 13 and 14 since $t = \Theta(n)$. ◀

3.3 Lower bound, malicious security, n parties, submaximal threshold

In this section we consider the case where t is submaximal, i.e. $n = 3t + s$ for some integer $s > 0$.

▶ **Theorem 16.** *There is a Boolean circuit C with g gates such that any reactive n-party protocol that computes C with perfect malicious security where $n = 3t + s$ for some $s > 0$, and t is the number of corruptions, must use total communication $\Omega(ng/s)$.*

Proof. By Lemma 13 it suffices to show a total communication lower bound of $\Omega(ntI/s)$. Consider any partition of the $2t+s$ honest parties into sets of size s. For simplicity assume s divides $2t+s$ so that any such partition consists of exactly $2t/s+1$ sets. We may group each set of s honest parties into a single party which we will call P_4. The remaining $3t$ parties may be arbitrarily grouped together into 3 groups of t parties each. This immediately gives a protocol for $\mathrm{IP}_{tI,4}$ where Theorem 11 applies, meaning P_4 must communicate $\Omega(tI)$. Since each set of k honest parties are disjoint we may add their communications together to get the total communication up to a constant factor. There are $2t/s+1$ such sets giving a total communication of $(2t/s+1)\Omega(tI) = \Omega(ntI/s) = \Omega(ng/s)$. ◀

3.4 Lower bound, malicious security, arithmetic circuits

The argument presented in previous sections only considered Boolean circuits, however the same argument applies to arithmetic circuits. Let $\mathbb{F}$ be a finite field whose elements require κ bits to describe. The exact same line of reasoning applies with the difference that $H(X_1) = \kappa I$ instead of $H(X_1) = I$. This increases the bounds by a factor of κ showing the following:

▶ **Theorem 17.** *There is an arithmetic circuit C with elements of size κ with g gates such that any reactive n-party protocol that securely computes C where $n = 3t + s$ for some $s > 0$, and t is the number of corruptions, must use total communication $\Omega(ng\kappa/s)$.*

3.5 Weakening the assumptions

Instead of assuming UC security we can instead make do with much weaker assumptions in order to show our lower bounds: What we can assume is a two-phase protocol as defined before, but with much weaker demands on the simulator than what we need for UC security, as we now sketch:

- The protocol in question can be split in two phases: we call the first one the input phase and the second the computation phase.
- The simulator first simulates the input phase and then the computation phase. It may rewind the adversary during both phases, but once it has started simulating the computation phase, it is not allowed to rewind back to the input phase.
- Once the simulator starts simulating the computation phase, and the functionality has received all the inputs, the simulator may now ask for the outputs (so this means it cannot ask for the output during the input phase).

It is not hard to see that our lower bound proofs go through, also in this model.

3.6 Lower bound, malicious security, statistical security

The lower bound presented above crucially relied on perfect security of the underlying protocol. In this section we briefly sketch where the lower bound for four parties breaks down in the case of statistical security. We show how the four parties may compute the function $\mathrm{IP}_{I,4}$ in a way where P_4 has a communication complexity of $O(\mathrm{poly}(n))$. In particular, the communication complexity of P_4 is independent of I, the input size.

It is well-known that we can compute any circuit with statistical security in an honest majority setting given access to a broadcast channel. We will then let $\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3$ run such a protocol, letting P_4 assist only in the broadcasts (since $t < n/3$ is required for broadcast). Specifically, P_4 produces a VSS of her input and broadcasts to the other parties, who then compute a VSS of P_4s output and sends back. We use the protocol by [1]. We denote by $X + Y \cdot \mathcal{BC}$ a communication complexity of X bits, and Y bits for broadcast.

▶ **Theorem 18** (Ben-Sasson, Fehr, Ostrovsky). *Let C be a Boolean circuit with g gates. Then there is a statistically secure MPC protocol (with security parameter κ) for computing C with communication complexity $O((n \lg n)\, g) + O(n^3\, \kappa) \cdot \mathcal{BC}$.*

The communication required for P_4 is dominated by the cost of doing broadcasts, which in particular is independent of I, the input size. This means the lower bound of Theorem 1 does not apply in the statistical setting, even without a broadcast channel. Interestingly, this suggests a “gap” between the two worlds.

We now show various upper bounds and compare them to the corresponding lower bounds. In most cases we are able to match the lower bounds up to a constant factor, however there is a gap of $O(\lg n)$ in the case of “unshaped” Boolean circuits, resulting from the fact that we need $> n$ evaluation points to do secret sharing.

3.7 Upper bound, malicious security, arithmetic circuits

For arithmetic circuits over large fields the parties can secret share their inputs and compute the circuit using Beaver triples. A recent protocol by [14] gives a protocol that is not dependent on the depth of the circuit being computed:

▶ **Theorem 19** (Goyal, Liu, Song). *If C is an arithmetic circuit with g gates over a field $\mathbb{F}$ with $|\mathbb{F}| > n$, and κ is the size of field element, then there is a perfect maliciously secure protocol for computing C using $O(ng\kappa + n^3\kappa)$ bits of communication.*

This shows that our lower bound of $\Omega(ng\kappa)$ is tight wrt. the circuit size for arithmetic circuits over fields of sufficient size. It also shows that our lower bound is the best generic lower bound one can hope to prove.

3.8 Upper bound, malicious security, $\mathrm{IP}_{I,n}$

The protocol from [14] is based on secret sharings and as a result requires fields with a size greater than the number of players, i.e. it must be the case that $|\mathbb{F}| > n$. This is because n distinct evaluation points are needed for the secret sharing. For smaller fields this is usually remedied by mapping elements into an extension field $\mathbb{K}$ and performing the secret sharings there. This unfortunately incurs an overhead of $O(\lg n)$ compared to our lower bound.

To remedy this for our specific function $\mathrm{IP}_{I,n}$ we can use *reverse multiplication friendly embeddings* (RMFE) following the work of [4]. An RMFE allows us to evaluate multiple small circuits in an extension field in parallel with good amortization in the communication.

▶ **Definition 20.** *Let $\mathbb{F}$ be a finite field. A k-RMFE scheme (ϕ, ψ) consists of two $\mathbb{F}$-linear mappings, $\phi : \mathbb{F}^k \to \mathbb{K}$, and $\psi : \mathbb{K} \to \mathbb{F}^k$ where for any vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}^k$ it holds that:*

$$\psi(\phi(\mathbf{a}) \cdot \phi(\mathbf{b})) = \mathbf{a} * \mathbf{b}$$

where $*$ is the coordinate-wise (Schur) product. This allows us to perform k parallel multiplications in $\mathbb{F}$ using a single multiplication in $\mathbb{K}$. Using an RMFE scheme, [4] construct a protocol for Boolean circuits with an amortized communication complexity of $O(n)$ per multiplication gate:

▶ **Theorem 21** (Cascudo et al.). *There is a secure n-party protocol for computing $\Omega(\lg n)$ parallel evaluations of a Boolean circuit with an amortized communication complexity of $O(n)$ per multiplication gate.*

We now establish an upper bound for our functionality $\mathrm{IP}_{I,n}$:

▶ **Theorem 22.** *There is a perfect maliciously secure protocol based on secret sharing for computing $IP_{I,n}$ using $O(n^2 I)$ bits of communication.*

Proof. Let C be the circuit described in Lemma 13. Assume for simplicity that $nI = 2^k$ and let $u = \Theta(k)$ be the the number of bits required to describe an element in $\mathbb{K}$. At a high level our strategy is to compose C into smaller circuits for which we get good amortization. The resulting computation is then computed without embeddings, in the hope that so much work was saved by parallelization that the remaining computation is asymptotically small.

The protocol is parameterized by an integer i that denotes the *depth* at which C is composed into smaller circuits: the parties first invoke the protocol from [4] until all but the last i layers remain, and then ignore the output reconstruction phase. At this point the parties have secret sharings of an element $w \in \mathbb{K}$ that encodes all 2^i wire values. The next step is extracting secret shares of each wire value. To do so, the parties generate sharings of random bits $[r_1], \ldots [r_u]$, encoding an element $[r]$ for some random $r \in \mathbb{K}$. To do this, each party contributes a random bit $[b]$ which are XORed together. To verify that the parties actually input bits, a public opening of $b^2 - b$ is produced and verified to equal 0 (as the only roots are 0 and 1). Next the parties compute $w - r$ and open the result in public. The result is added to $[r]$ to get a sharing $[w]$. Linearity of the secret sharing implies the parties may apply ψ locally to get a secret sharing of each wire value. Finally the parties invoke the protocol [7] on the shares obtained on the rest of the circuit.

Let $i = \Theta(\lg n)$ and let us analyze the communication complexity. It is clear that the cost is dominated by the first phase since the remaining two steps do not depend on I. It is also clear that the size of the circuit is $\Theta(nI)$ since there are nI inputs. By Theorem 21 the complexity of the first phase is $O(n) \cdot nI = O(n^2 I)$ as we wanted to show. ◀

3.9 Upper bound, malicious security, submaximal threshold

Both of the previous upper bounds assumed a maximal threshold of $n = 3t + 1$. In this section we briefly consider the case of submaximal threshold, i.e. where $n = 3t + s$ for some $s > 1$. In this setting we can use *packed secret sharing* to "pack together" s shares into a single element, allowing us to evaluate multiple gates in parallel and saving a factor $O(s)$ in communication. This matches the submaximal lower bound shown in this paper up to a constant factor. This shows that packed secret sharing is the best kind of optimization in terms of communication one could hope to achieve.

4 Conclusion and future work

In this paper we showed two classes of lower bounds for information-theoretic multiparty computation. For the case of active security, we have show the bound for a reactive functionality. It remains open whether a similar bound can be shown for (non-reactive) secure function evaluation.

References

1 Eli Ben-Sasson, Serge Fehr, and Rafail Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 663–680, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

2 Carlo Blundo, Alfredo De Santis, Giuseppe Persiano, and Ugo Vaccaro. Randomness complexity of private computation. *Computational Complexity*, 8(2):145–168, 1999.

3 R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145, 2001. doi:10.1109/SFCS.2001.959888.

4 Ignacio Cascudo, Ronald Cramer, Chaoping Xing, and Chen Yuan. Amortized complexity of information-theoretically secure mpc revisited. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 395–426, Cham, 2018. Springer International Publishing.

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

6 Ivan Damgård, Kasper Green Larsen, and Jesper Buus Nielsen. Communication lower bounds for statistically secure mpc, with or without preprocessing. In *CRYPTO (2)*, volume 11693 of *Lecture Notes in Computer Science*, pages 61–84. Springer, 2019.

7 Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. In *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 572–590. Springer, 2007.

8 Ivan Damgård, Jesper Buus Nielsen, Rafail Ostrovsky, and Adi Rosén. Unconditionally secure computation with reduced interaction. In *EUROCRYPT (2)*, volume 9666 of *Lecture Notes in Computer Science*, pages 420–447. Springer, 2016.

9 Ivan Damgård, Jesper Buus Nielsen, Antigoni Polychroniadou, and Michael A. Raskin. On the communication required for unconditionally secure multiplication. In *CRYPTO (2)*, volume 9815 of *Lecture Notes in Computer Science*, pages 459–488. Springer, 2016.

10 Deepesh Data, Manoj M. Prabhakaran, and Vinod M. Prabhakaran. On the communication complexity of secure computation. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014*, pages 199–216, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

11 Uri Feige, Joe Killian, and Moni Naor. A minimal model for secure computation (extended abstract). In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '94, page 554–563, New York, NY, USA, 1994. Association for Computing Machinery. doi:10.1145/195058.195408.

12 Matthew Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '92, page 699–710, New York, NY, USA, 1992. Association for Computing Machinery. doi:10.1145/129712.129780.

13 Anna Gal and Adi Rosen. Lower bounds on the amount of randomness in private computation. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '03, page 659–666, New York, NY, USA, 2003. Association for Computing Machinery. doi:10.1145/780542.780638.

14 Vipul Goyal, Yanyi Liu, and Yifan Song. Communication-efficient unconditional mpc with guaranteed output delivery. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 85–114, Cham, 2019. Springer International Publishing.

15 Eyal Kushilevitz and Yishay Mansour. Randomness in private computations. In *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '96, page 181–190, New York, NY, USA, 1996. Association for Computing Machinery. doi:10.1145/248052.248089.

16 Eyal Kushilevitz and Adi Rosén. A randomness-rounds tradeoff in private computation. In Yvo G. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, pages 397–410, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

17 I. Parberry. The pairwise sorting network. *Parallel Process. Lett.*, 2:205–211, 1992.