Communication Lower Bounds for Perfect Maliciously Secure MPC

Ivan Damgård* and Nikolaj I. Schwartzbach*

Dept. of Computer Science, Aarhus University

Abstract. We prove a lower bound on the communication complexity of perfect maliciously secure multiparty computation, in the standard model with $n = 3t + 1$ parties of which t are corrupted. We show that for any n and all large enough $g \in \mathbb{N}$ there exists a Boolean circuit C with g gates, where any perfectly secure protocol implementing C must communicate $\Omega(ng)$ bits. The results easily extends to constructing similar circuits over any fixed finite field. Our results also extend to the case where the threshold t is suboptimal. Namely if $n = 3t+s$ the bound is $\Omega(ng/s)$, which corresponds to known optimizations via packed secret-sharing. 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).

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 her 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

* Supported by the ERC Advanced Grant MPCPRO.

relevant existing work for us is [DLN19] 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 some functions. 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 one important case, namely lower bounds for protocols that achieve *perfect malicious* security. It is well known that this can be done if and only if $t < n/3$ and the result from [DLN19] has nothing to say about this case: to apply it, one would need to set s to be $\Theta(n)$ and then the lower bound becomes trivial. Thus the open question we consider is: *can we show lower bounds for perfect malicious security in the case where $n = 3t + 1$?*

1.1 Our results

In this paper, we prove lower bounds for the model with n parties of which t are actively and 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 perfect and maliciously secure protocols with static corruption in the model where $n = 3t+1$. We require that protocols are UC secure and have a standard 2-phase structure where the inputs are committed in the first phase, before the output is computed in the second phase. All known perfectly secure protocols have this structure, which is natural since malicious security requires that corrupt parties cannot choose their inputs as a function of the honest parties' inputs. So, nothing can be known about the output when a corrupt party chooses its input. Thus the only restriction we require is that there must be some explicitly defined point in the protocol where all inputs have been committed to. We assume UC security mainly for simplicity of exposition, we can actually make do with significantly weaker assumptions, this is detailed in Section 2.5.

As mentioned, any function can be computed insecurely by sending the inputs to one party and let her compute the function. This takes communication $O(S)$ where S is the input size (recall that we assume that the overall size of the output is smaller than S). What we show is now that for all n and any sufficiently large S, there exists a function f with input size S such that any protocol that evaluates f securely must communicate $\Omega(nS)$ bits. So this answers the above open question.

Even more is true: we are able to construct functions f 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, where any protocol that evaluates C securely must communicate $\Omega(ng)$ bits. The result easily extends to constructing similar circuits over any fixed finite field.

We emphasize that our result only talks about functions with linear size circuits, so this 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.

Our results extend 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.

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 [GLS19] is a factor $\lg n$ off for Boolean circuits. We do this by exploiting recent results by Cascudo et al. [CCXY18] 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.

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.

To show our results, we start from a lower bound for the communication complexity of a specific function for the case of 4 parties and 1 maliciously corrupt player. We then "lift" this result to the multiparty case. This high-level strategy is similar the one used in [DLN19], however our proof for the 4-party case as well as the concrete lifting technique are very different from what was done in [DLN19]. In fact it is easy to see that new techniques are necessary to achieve our result. Namely, in our case where $t < n/3$, [DLN19] only gives a trivial result, as mentioned above. Nevertheless [DLN19] 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 [Kus92, FY92, CK93, FKN94, KM97, KR94, BSPV99, GR03] (and see [DPP14] for an overview of these results). The previous work most relevant to us is [DPP14]. 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 [DNOR16], 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 [DNOR16] is not interesting in our context.

In [DNPR16], 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

In this section we prove that there is an n-party function describable 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 $\mathsf{P}_1, \ldots \mathsf{P}_n$ 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 allow f to be probabilistic.

We will consider protocols that implement the standard ideal functionality F_f for computing f securely, namely the one that receives input from all parties, and once all inputs have been contributed, it computes the function and sends the output to all parties.

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 ([Can00]). We assume the reader is familiar with the model and its definition of security:

Definition 1 (UC Security). *A protocol π is said to securely realize a functionality F with perfect malicious security if for any adversary $\mathcal{A}$ there exists a simulator S such that $\pi \diamond \mathcal{A}$ is perfectly indistinguishable from $S \diamond \mathsf{F}$.*

Finally, we will consider *two-phase* protocols that can be split in two phases: we call the first one the input phase and the second the computation phase. The idea is that the inputs are committed in the first phase, before any information on the output is revealed. This is defined more precisely in terms of the simulation:

the split in phases must be such that when simulation of the input phase is over, the functionality has received all the inputs, but has not sent the output yet[1].

2.1 Lower Bound, Perfect Security, Four Parties

We start by considering a special case of active MPC with four parties $\mathsf{P}_1, \ldots \mathsf{P}_4$. The functionality first receives an input $X_i \in \{0,1\}^I$ from each P_i. After having received all four inputs it computes the result and sends the result to each party. We denote by $\mathrm{IP}_{I,4}$ the function that chooses a random index $i \in_R [0, 4I)$ and outputs $(i, \mathbf{X}_i)$, where $\mathbf{X} = X_1||X_2||X_3||X_4$ is the concatenation of the party inputs. The function $IP_{I,4}$ has the crucial property that the output distribution changes whenever any party changes their input.

One consequence of this is the following lemma:

Lemma 1. *Assume protocol π computes n-party probabilistic function f with perfect security, and it is the case that for any $x_1, ..., x_n$ and $x'_1 \neq x_1$, the output distribution of $f(x_1, ..., x_n)$ is different from that of $f(x'_1, x_2, ..., x_n)$. 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 distribution must be that of $f(x_1, ..., x_n)$. If instead P_1 is corrupt but plays honestly, the protocol does exactly the same as if all players are honest so the output distribution is still that of $f(x_1, ..., x_n)$. Hence, when simulating this case, the simulator must send x_1 to the functionality, otherwise the output distribution generated in the simulation will be incorrect, 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 1. *In any two-phase protocol that implements $\mathsf{F}_{\mathrm{IP}_{I,4}}$ 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

[1] Note that UC simulation is always straight-line, so it makes to talk about the point where simulation of a phase is over.

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:

$$\mathrm{H}(X_1 \mid T_2) = \mathrm{H}(X_1) \tag{1}$$

$$\mathrm{H}(X_1 \mid T_2, T_4) = 0 \tag{2}$$

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 $x_1 \neq x'_1$. Now consider the following two attacks on π:
   (a) P_3 is corrupt, but 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 complete the protocol honestly, assuming that her view of the input phase was (x_3, r'_3, t'_3).
   (b) P_1 is corrupt but plays honestly in the input phase. If P_1 received input x'_1 and 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).

   We can now observe that when the real protocol executes in the first scenario, 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 scenario 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'_1, t'_1), (x_4, r_4, t_4), where all random coins are uniform, given the corresponding transcript. Since these views are identically distributed in the two cases 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 1. Since $x_1 \neq x'_1$, the two resulting output distributions D_1 and D_2 are different.
However, we have just seen that the real protocol may sometimes generate output distribution D under both the first and the second attack. Perfect security now implies that $D_1 = D = D_2$, a contradiction. □

2.2 Lower Bound, Perfect 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 collected in $\mathbf{X}$, a random index i is chosen and $(i, \mathbf{X}_i)$ outputted. This function has the following property: if any party changes their input, the output distribution will also change.

Lemma 2. *$\mathrm{IP}_{I,n}$ 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 $O(\lg S)$ *random bits* which we will denote by $\mathbf{r}$. We 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 3. *Any two-phase 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 $\mathrm{IP}_{tI,4}$ where P plays the role of P_4. Corrupting any party in $\mathrm{IP}_{tI,4}$ corrupts at most t parties in $\mathrm{IP}_{I,n}$. By Theorem 1, 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 2. *There is a (familiy of) Boolean circuit(s) C with g gates such that any two-phase n-party protocol computes C with perfect malicious security must use total communication $\Omega(ng)$.*

Proof. Follows immediately from Lemmas 2 and 3 since $t = \Theta(n)$. □

2.3 Lower Bound, Perfect 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 3. *There is a Boolean circuit C with g gates such that any two-phase 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 2 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 1 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 $(2t/s + 1)\Omega(tI) = \Omega(ntI/s) = \Omega(ng/s)$. □

2.4 Lower Bound, 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 4. *There is an arithmetic circuit C with elements of size κ with g gates such that any 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 $\Omega(ng\kappa/s)$.*

2.5 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, the functionality has received all the inputs, and 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 Upper Bounds

In this section we 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.

3.1 Upper Bound, 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 [GLS19] gives a protocol that is not dependent on the depth of the circuit being computed:

Theorem 5. *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.

3.2 Upper Bound, $\mathrm{IP}_{I,n}$

The protocol from [GLS19] 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 [CCXY18]. An RMFE allows us to evaluate multiple small circuits in an extension field in parallel with good amortization in the communication.

Definition 2. *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, [CCXY18] construct a protocol for Boolean circuits with an amortized communication complexity of $O(n)$ per multiplication gate:

Theorem 6. *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.*

Proof. See [CCXY18]. □

Theorem 7. *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 2. 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 [CCXY18] 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 [DN07] 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 6 the complexity of the first phase is $O(n) \cdot nI = O(n^2 I)$ as we wanted to show. □

3.3 Upper Bound, 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

We have shown that any generic protocol for perfect maliciously secure MPC must have a worst-case communication complexity linear in the number of players and the size of the circuit being computed. In particular, we constructed a class of circuits over any fixed finite field for which any n-party protocol must communicate $\Omega(ng\kappa)$ bits, where g is the size of the circuit being computed. For arithmetic circuits over finite fields of sufficient size, we have seen that the best known upper bound equals our lower bound up to a constant factor.

We showed that the particular class of circuits used in the lower bound can be computed with a communication that equals the lower bound up to a constant factor. We were able to do so because we could exploit the recursive structure of the circuits to amortize some communication. It is unclear how to apply this technique in general to "tall and skinny" Boolean circuits. As a consequence it is unlikely there is generic protocol for Boolean circuits with a communication complexity that equals our lower bound.

When the number of corruptions is submaximal, i.e. $n = 3t+s$ for some $s > 1$ we showed that the lower bound is $\Omega(ng/s)$. This equals the communication saved by the use of packed secret sharing which shows that packed secret sharing is optimal in some sense.

The specific class of circuits we considered is linear in the size of the input. The obvious open problem is whether the lower bound still applies when the circuit is much larger than the input: is there a lower bound that grows with the circuit size of the function being computed?

References

[BSPV99] Carlo Blundo, Alfredo De Santis, Giuseppe Persiano, and Ugo Vaccaro. Randomness complexity of private computation. *Computational Complexity*, 8(2):145–168, 1999.

[Can00] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. https://eprint.iacr.org/2000/067.

[CCXY18] Ignacio Cascudo, Ronald Cramer, Chaoping Xing, and Chen Yuan. Amortized complexity of information-theoretically secure MPC revisited. In *CRYPTO (3)*, volume 10993 of *Lecture Notes in Computer Science*, pages 395–426. Springer, 2018.

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

[DLN19] 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.

[DN07] 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.

[DNOR16] 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.

[DNPR16] 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.

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

[FKN94] Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). pages 554–563, 1994.

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

[GLS19] 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.

[GR03] Anna Gál and Adi Rosén. Lower bounds on the amount of randomness in private computation. pages 659–666, 2003.

[KM97] Eyal Kushilevitz and Yishay Mansour. Randomness in private computations. *SIAM J. Discrete Math.*, 10(4):647–661, 1997.

[KR94] Eyal Kushilevitz and Adi Rosén. A randomnesss-rounds tradeoff in private computation. pages 397–410, 1994.

[Kus92] Eyal Kushilevitz. Privacy and communication complexity. *SIAM J. Discrete Math.*, 5(2):273–284, 1992.