Unconditionally Secure Computation with Reduced Interaction

Ivan Damgård[1], Jesper Buus Nielsen[1], Rafail Ostrovsky[2], and Adi Rosén[3]

[1] Dept. of Computer Science, Aarhus University
[2] UCLA
[3] CNRS & Université Paris Diderot

Abstract. We study the question of how much interaction is needed for unconditionally secure multiparty computation. We first consider the number of messages that need to be sent to compute a Boolean function with semi-honest security, where all n parties learn the result. We consider two classes of functions called t-difficult and t-very difficult functions, where t refers to the number of corrupted players. For instance, the AND of an input bit from each player is t-very difficult while the XOR is t-difficult but not t-very difficult. We show lower bounds on the message complexity of both types of functions, considering two notions of message complexity called conservative and liberal, where conservative is the more standard one. In all cases the bounds are $\Omega(nt)$. We also show (almost) matching upper bounds for $t = 1$ and functions in a rich class PSM_{eff} including non-deterministic log-space, as well as a stronger upper bound for the XOR function. In particular, we find that the conservative message complexity of 1-very difficult functions in PSM_{eff} is $2n$, while the conservative message complexity for XOR (and $t = 1$) is $2n - 1$. Next, we consider round complexity. It is a long-standing open problem to determine whether all efficiently computable functions can also be efficiently computed in constant-round with *unconditional* security. Motivated by this, we consider the question of whether we can compute any function securely, while minimizing the interaction of *some of* the players? And if so, how many players can this apply to? Note that we still want the standard security guarantees (correctness, privacy, termination) and we consider the standard communication model with secure point-to-point channels. We answer the questions as follows: for passive security, with $n = 2t + 1$ players and t corruptions, up to t players can have minimal interaction, i.e., they send 1 message in the first round to each of the $t + 1$ remaining players and receive one message from each of them in the last round. Using our result on message complexity, we show that this is (unconditionally) optimal. For malicious security with $n = 3t + 1$ players and t corruptions, up to t players can have minimal interaction, and we show that this is also optimal.

1 Introduction

In Multiparty Computation n players want to compute an agreed-upon function on privately held inputs, such that the desired result is correctly computed and is the only new information released. This should hold even if t players have been actively or passively corrupted by an adversary.

If point-to-point secure channels between players are assumed, any function can be computed with unconditional (perfect) security, against a passive adversary if $n \geq 2t+1$ and against an active adversary if $n \geq 3t+1$.[BGW88,CCD87] If we assume a broadcast channel and accept a small error probability, $n \geq 2t+1$ is sufficient to get active security [RB89].

The protocols behind these results require a number of communication rounds that is proportional to the depth of an (arithmetic) circuit computing the function. One would of course like to compute any function with unconditional security, in constant rounds, and efficiently in terms of the circuit size of the function. This is however a long-standing open problem (note that this is indeed possible if one makes computational assumptions).

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

It is therefore very natural to ask whether this state of affairs is inherent. How much interaction do we actually need for unconditional security, and can we reduce the interaction needed compared to existing protocols? This type of question was studied in [FKN94,DPP14] in a specific 3-party model where 2 parties have input and a third gets the output. We further detail below some previous work on secure addition, but in general very little is known on this question.

In this paper, we make some progress with respect to two related but different measures of interaction: message complexity and round complexity, in the context of synchronous networks.

Message complexity seems like a very simple measure at first sight: simply count how many messages are sent in the protocol. However, a moment's thought will show that things are a bit more tricky. For instance, what if the protocol varies its communication pattern, so that P_i sometimes (but not always) sends a message to P_j in a certain round? One way to handle this is to declare that the absence of a message is also a signal. This leads to what we call *conservative message complexity*, i.e., we say that if P_i sometimes sends a message to P_j in a certain round, then we consider it to be the case that P_i always sends a message to P_j in this round. This way, we force protocols to have a fixed communication pattern.

However, considering only this measure is not completely satisfying. After all, it could be that one could design protocols with a smaller number of messages by using tricks such as waiting for a certain time before a message is sent, and using the amount of elapsed time as an implicit signal. In real life such an approach could be interesting, as there may be some cost involved in physically moving a message, that is not incurred if one stays silent. Therefore, we also define *liberal message complexity*, where the protocol is only charged for messages that are *explicitly* sent, and where we consider the *expected* number of messages as well the maximum. We discuss these measures in more detail later, when we define them formally.

Our results are as follows: We consider n players and t semi-honest and static corruptions. We look at statistically secure computation of Boolean functions, where all parties learn the output. We assume secure point to point channels that leak the length of the message sent to the adversary (as any implementation using crypto would do). The ideal functionality for computing the function leaks the output to the adversary only if some party is corrupted, so essentially we ask that the adversary cannot learn anything by doing only traffic analysis.

We consider two classes of functions, called t-very difficult and a larger class called t-difficult. The AND of an input bit from each player, and more generally threshold functions are t-very difficult, whereas the XOR is t-difficult but not t-very difficult.

We show lower bounds for all 4 cases that arise naturally. In all cases the bounds are $\Omega(nt)$. Results are summarized in Figure 1.

	Liberal	Conservative
t-very difficult	$\lceil\frac{n(t+1)-1}{2}\rceil + \frac{n}{2}$	$\lceil\frac{n(t+1)-1}{2}\rceil + n$
t-difficult	$\lceil\frac{n(t+1)-1}{2}\rceil + \frac{n-1}{2}$	$\lceil\frac{n(t+1)-1}{2}\rceil + n - 1$

Fig. 1. Lower Bounds.

For the case of $t = 1$ we also show upper bounds using perfectly secure protocols, for all functions in a class we call PSM_{eff} which includes non-deterministic log-space and more (see Definition 1 below), as well as a stronger upper bound for the XOR function. Figures 2 and 3 show the lower bounds for $t = 1$ and the upper bounds. We see that we have obtained the exact conservative message complexity for all 1-very difficult functions in PSM_{eff}. This includes, for instance, the AND and thresholds functions in general. We have also obtained the exact conservative and liberal message complexity for XOR (when $t = 1$). Finally we have characterised the liberal message complexity of 1-very difficult functions in PSM_{eff} up to 1/2 message, the exact characterization is left as an open problem.

	Liberal	Conservative
1-very difficult	$3n/2$	$2n$
1-difficult	$3n/2 - 1/2$	$2n - 1$

Fig. 2. Lower Bounds for $t = 1$.

	Liberal	Conservative
PSM_{eff}	$3n/2 + 1/2$	$2n$
XOR	$3n/2 - 1/2$	$2n - 1$

Fig. 3. Upper Bounds for $t = 1$.

Some remarks on alternative models are in order: we insist that the number of parties is considered to be constant, even if the security parameter grows. This rules out tricks like secret sharing one's input among a small subset of parties, hoping they are not all corrupt [BGT13,GIPR] (which works for static corruptions, but not for adaptive corruptions). If one is happy with statistical, static, semi-honest security for a large number of parties, then this type of trick can be used to compute simple operations with a poly-log (in n) number of messages. If the communication pattern is fixed, than a quadratic number of messages is required for addition protocols [CK93]. Note that our bounds hold regardless of the number of parties if adaptive security or perfect security is required (and our upper bounds yield perfect security). Therefore the only way to circumvent our lower bounds is to settle for static and statistical security and let the number of parties grow with the security parameter (for adaptive adversary with setup assumptions, see further discussion in [CCG$^+$15]).

Next, we consider round complexity: As mentioned, computing any function with unconditional security, in constant rounds and efficiently in the circuit size of the function is an open problem[4], and providing a positive answer seems to require completely new ideas for protocol design. Motivated by this, we consider the question of whether we can minimize the interaction of *some of* the players? And if so, how many players can this apply to? Note that we still want the standard security guarantees (correctness, privacy, termination). We answer this question as follows: for passive security, with $n = 2t+1$ players and t corruptions, up to t players can have minimal interaction, i.e., they send 1 message in the first round to each of the $t + 1$ remaining players and receive one message from each of them in the last round. Using our result on message complexity, we show that this is (unconditionally) optimal. For malicious security with $n = 3t + 1$ players and t corruptions, up to t players can have minimal interaction, and we show that this is also optimal.

For the purpose of proving the positive result for malicious security, we show a result of independent interest: For the case $n = 3t+1$ and t malicious corruptions, we design a broadcast protocol of the following special form: we can select any subset of t players, who only need to send one message to the other $n-t$ players. After this point, we can do broadcast among the remaining $n - t$ players. Note that we are not guaranteed that we have at most a third corruptions among the $n - t$ players, so we cannot do broadcast from scratch in this set. We find it

[4] Using randomizing polynomials [IK00] one can get unconditional security and constant round efficiently in the branching program size of the function, but this does not seem to help beyond NC^1.

slightly surprising that we need so little involvement from the t selected players. In particular, they might all be corrupt and hence send completely inconsistent setup values – then, of course, we are saved by the fact that the remaining players are all honest (but they do not know this yet).

2 Preliminaries

We use $\mathbb{N}$ to denote the non-negative integers. For $n \in \mathbb{N}$ we let $[n] = \{1, \ldots, n\}$.

We prove security in the model from [Can00] with unconditional security and a static adversary. We consider a synchronous model with point-to-point perfectly secure channels between each pair of parties, where the length of each message sent is leaked to the adversary. In one round, all parties may send messages to each other. We consider function evaluation between n parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ with inputs $x_1, \ldots, x_n$ and common output $y = f(x_1, \ldots, x_n)$ for a poly-time n-party function f. In the ideal model, we assume that nothing is leaked to the adversary in case no one is corrupted. We refer to [Can00] for the details of the model.

We say that a protocol has perfect correctness if it always computes the correct result when all parties follow the protocol. We say that a protocol has perfect privacy against t semi-honest corruptions if the ideal world and the real world models have the same distributions even when t parties are passively corrupted, i.e., they follow the protocol but might pool their views of the protocol to learn more than they should. We say that a protocol has statistical privacy against t semi-honest corruptions if the view of the corrupted parties in the ideal world and the real world models have distributions that are statistically close in some security parameter s even if t parties are passively corrupted. We say that a protocol has perfect privacy against t malicious corruptions if the view of the corrupted parties in the ideal world and the real world models have the same distributions even when t parties might deviated from the protocol in a coordinated manner. If the distributions are only statistically close we talk about statistical security against t malicious corruptions.

As is well known, it is possible to implement secure function evaluation of any poly-time n-party function with perfect correctness and perfect privacy against t semi-honest corruptions when $n \geq 2t + 1$. It is possible to implement secure function evaluation of any poly-time n-party function with perfect correctness and perfect privacy against t malicious corruptions when $n \geq 3t + 1$, see [BGW88,CCD87].

We will use secure function evaluation protocols for the so-called preprocessing model as tools. In these protocols an incorruptible trusted third party will sample a distribution D to get an n-tuple $(d_1, \ldots, d_n) \leftarrow D$. Then it privately gives d_i to P_i. After the setup phase, the n parties engage in a protocol where they communicate over secure channels. In such pre-processing models there exist appropriate distributions D which will allow to get perfect correctness and perfect privacy against t passive corruptions out of $n = t + 1$ parties. See, e.g., [DZ13] and the references therein.

We also use protocols for the private simultaneous message (PSM) model. For this model an n-party protocol for an n-party function f is given by

$$(R, M_1, \dots, M_n, g) \ ,$$

where R is a distribution with finite support, each M_i is a function, called the message function of party i, and g is function called the reconstruction function.

By perfect correctness of a PSM protocol for an n-party function f we mean that for all r in the support of R and all inputs $(x_1, \dots, x_n)$ for f it holds that $f(x_1, \dots, x_n) = g(M_1(x_1, r), \dots, M_n(x_n, r))$.

By ϵ-privacy of a PSM we mean that there exists a poly-time simulator S such that for all inputs $(x_1, \dots, x_n)$ for f, $y = f(x_1, \dots, x_n)$ and a random sample $r \leftarrow R$ it holds that $(M_1(x_1, r), \dots, M_n(x_n, r))$ and $S(y)$ have statistical distance at most ϵ. If $\epsilon = 0$, then we talk about perfect privacy. If ϵ is negligible we talk about statistical security. Privacy ensures that a party seeing $(M_1(x_1, r), \dots, M_n(x_n, r))$ learns nothing extra to $y = g(M_1(x_1, r), \dots, M_n(x_n, r))$.

The PSM model is a generalization of [FKN94] and is defined in [IK97], where they also gave perfectly secure and efficient (poly-time) PSM protocols for a large class of functions including non-deterministic log-space, $\mathrm{mod}_p\,\mathrm{L}$ and $\sharp\,\mathrm{L}$. In [IK97] privacy is not formulated via poly-time simulation: the notion only asks that $(M_1(x_1, d_1), \dots, M_n(x_n, d_n))$ depends only on $f(x_1, \dots, x_n)$. We need the simulation based notion here, as we prove security in [Can00], which is phrased via efficient simulation. We note that if for a given function f it is possible to compute in poly-time, from an output $y = f(x_1, \dots, x_n)$, an input $(x'_1, \dots, x'_n)$ such that $y = f(x'_1, \dots, x'_n)$ then the notions are equivalent for f. The simulator will simply compute $(x'_1, \dots, x'_n)$, sample $r \leftarrow R$ and output $(M_1(x_1, r), \dots, M_n(x_n, r))$. Of course, if inputs are single bits and the number of parties is considered to be constant, such inversion can be done in constant time by trying all possibilities.

In the following, when using PSM protocols, we will consider such efficiently invertible functions f that also have an efficient PSM protocol:

Definition 1. *We will use PSM_{eff} to denote the class of functions that are efficiently invertible as described above and can be computed by a polynomial time PSM protocol.*

We also use additive secret sharing of bits strings $x \in \{0,1\}^m$. An additive secret sharings of x between $\mathsf{P}_1, \dots, \mathsf{P}_n$ consists of sampling shares $s_1, \dots, s_n \in (\{0,1\}^m)^n$ uniformly at random under the only restriction that $x = \oplus_{i=1}^n s_i$, where $\oplus$ denote bit-wise exclusive or. It is easy to show that the distribution of any $n-1$ of the shares is the uniform one on $(\{0,1\}^m)^{n-1}$ and hence independent of x.

3 Message Complexity

Defining the message complexity of a protocol for the synchronous model with secure channels appropriately is slightly more tricky than one might expect at first, so we address this issue in its own section.

We will first of all need to allow parties to *not* send a message to some party in a given round. Since all parties send messages to all parties in all rounds in [Can00], we need to hack the model a bit for this. We will say that if a party sends the empty string then this counts as not having sent a message. Think of receiving the empty string from P_i as meaning "no message was received from P_i in this round".

This builds up to a subtler point that we demonstrate by an example. Consider the problem where a dealer D is to deal an additive secret sharing of a bit d between n parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$. What is the average message complexity of this problem? It turns out that if we ignore security for a second, then it is at most $n/2$ if one is not careful. The dealer samples a secret sharing $d = d_1 \oplus \cdots \oplus d_n$. Then for $i = 1, \ldots, n$, if $d_i = 0$ he does not send a message to P_i. If $d_i = 1$, then he sends 1 to P_i. Since d_i is uniformly random it follows from linearity of expectation that he sends an expected $n/2$ messages.

If we consider security, the bound changes. It is the case in [Can00] that the adversary can see the length of a message sent securely. This in particular means that in our setting here, the adversary can see if a message was sent or not between any two parties—it can see the communication pattern. This is a reasonable model, as hiding the presence of a communication is not practical, in particular when we actually do not want to transmit anything when there is no message to be sent.

Of course seeing the communication pattern of the above protocol renders it insecure, but this kind of contrived example shows that in some cases, if we want a very precise measure of message complexity we need to consider protocols with fixed communication patterns, i.e., if P_1 sometimes sends a message to P_2 in round 1, then we consider it the case that P_1 always sends a message to P_2 in round 1, as the absence of the message is a signal.

On the other hand, considering only this measure seems to be not entirely satisfying. We should be intrigued whether or not using tricks as above will allow more efficient protocols, so it makes sense to also consider a notion where we only count messages that are *explicitly* sent.

This will mean that the number of messages may not be the same in all runs of the protocol. When we prove lower bounds it will therefore not be meaningful to consider conservative message complexity. For example, if we can prove that all protocols must with some probability 2^{-s}, where s is the security parameter, send $2^{40}n$ message but that they in all other cases might have to send only $2n$ messages, then we would not consider $2^{40}n$ a very meaningful *lower bound* for the number of messages. When we prove lower bounds we would like to consider expected message complexity, which would turn the lower bound in the just given example into $2n$, as $2^{-s}2^{40}n$ is vanishing in s. We call this liberal communication complexity. Another way to relax the conservative notion is to still only count messages explicitly sent but look at the worst case number over the randomness of the parties. We call this worst case communication complexity. It is obviously in between the conservative and liberal notions and we will at some point only be able to prove an upper bound for the worst case notion (as opposed to the conservative one).

We therefore define three measures of message complexity, a conservative one, a liberal one and a worst case one:

Definition 2 (Conservative Message Complexity). *Let π be an n-party protocol for a synchronous network. Let R be random tapes of all players. By $\mathsf{Msg}_{\texttt{con}}(\pi)$ we denote the conservative message complexity of π. For all $r \in \mathbb{N}$ and all $i \in [n]$ and all $j \in [n] \setminus \{i\}$ we define $c_{r,i,j}$ to be 1 if there exists an input x for π and randomness R such that when π is run with that input and that randomness, P_i will send a message to P_j in round r. We let $c_{r,i,j} = 0$ otherwise. We let*

$$\mathsf{Msg}_{\texttt{con}}(\pi) = \sum_{r,i,j} c_{r,i,j} \ .$$

Note that in the conservative message complexity, even if some player flips a fair coin and sends a message that is independent of it's input, say "hello" to player one if the coin is zero and "hello" to player two, three and four if the coin is one, the conservative message complexity counts this as four messages. A more liberal way to count messages in any specific protocol run and then take expectation or worst case over the random tapes of the parties. We call this liberal message complexity respectively worst case complexity. In the above example, the liberal message complexity of the "hello" messages is two messages and the worst case complexity is three message.

Definition 3 (Liberal Average/Worst-case Message Complexity). *Let π be an n-party protocol for a synchronous network. For a given run of π on input $\boldsymbol{x}$ and some fixed random tapes $\boldsymbol{R}$ of the parties we define $c_{r,i,j}$ to be 1 if P_i sent a message to P_j in round r. We let $c_{r,i,j} = 0$ otherwise. We let*

$$\mathsf{Msg}(\pi, \boldsymbol{x}, R) = \sum_{r,i,j} c_{r,i,j}$$

and

$$\mathsf{Msg}_{\texttt{lib}}(\pi) = \max_{\boldsymbol{x}} \mathrm{E}_{\boldsymbol{R}}[\mathsf{Msg}(\pi, \boldsymbol{x}, \boldsymbol{R})] \ .$$

$$\mathsf{Msg}_{\texttt{wor}}(\pi) = \max_{\boldsymbol{x},\boldsymbol{R}}[\mathsf{Msg}(\pi, \boldsymbol{x}, \boldsymbol{R})] \ .$$

It is easy to see that it is always the case that $\mathsf{Msg}_{\texttt{lib}}(\pi) \leq \mathsf{Msg}_{\texttt{wor}}(\pi) \leq \mathsf{Msg}_{\texttt{con}}(\pi)$.

We extend the above notions to the statistical setting by defining them as above for each fixed value of σ and then taking $\limsup$ when this limit is defined. If this limit is not defined, we define the message complexity to be ∞.

4 Lower Bounds

We now proceed to present and prove our lower bounds. We first prove a lower bound on the message complexity of secure function evaluation in the face of semi-honest corruptions. Then we give a lower bound on the individual round complexity in the face of t semi-honest corruptions and then a lower bound on the individual round complexity in the face of t malicious corruptions.

4.1 Message Complexity

We first prove a lower bound on the message complexity of secure function evaluation secure against t semi-honest corruptions. We will prove the bound for a large class of function that we will call t-difficult, and a slightly larger bound for a smaller class called t-very difficult.

First some clarifications: even though we have defined two different ways to count messages, where an empty message counts in one notion and not in the other, in the following, when we say that a message is sent or received, or messages are exchanged, we always refer to non-empty messages.

Very roughly, the intuition we will formalize is as follows: A player whose input matters to the result must somehow communicate his input to the rest of players, in order to enable correct computation of the result by all players. The input cannot be encoded in the communication pattern which is public, so it must follow from the content of messages this player exchanges with other players. On the other hand, a player whose inputs matters has to exchange messages with at least $t + 1$ parties before his input becomes determined. Otherwise he may have talked to only corrupted parties and the protocol would not be private. This already indicates a lower bound of $n(t + 1)/2$ messages (we need to divide by 2 since a message counts as communication for both sender and receiver). But we can do more: we show that *after* the inputs have been fixed, all players must receive information allowing them to determine the result of the computation. Under the liberal message complexity notion, this does not necessarily mean that all players must receive another message, but we can show that in expectation most players must receive a message half the time. So this indicates a lower bound of $n(t + 2)/2$ messages, which is (approximately) what we obtain.

We start with some notation: For an input vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ and a subset $D \subseteq \{1, \ldots, n\}$ and inputs $x_D = \{(j, x'_j)\}_{j \in D}$ for the parties in D we use $\boldsymbol{x}[x_D]$ to denote the vector $\boldsymbol{x}$ with x_j replaced by x'_j for $j \in D$.

Definition 4. *We say that a function f is* t-difficult for P_i *if the following holds:*

influence *There exists two inputs $\boldsymbol{x}^{i,0}$ and $\boldsymbol{x}^{i,1}$ such that $\boldsymbol{x}^{i,0}_j = \boldsymbol{x}^{i,1}_j$ for all $\mathsf{P}_j \neq \mathsf{P}_i$ and such that $f(\boldsymbol{x}^{i,0}) \neq f(\boldsymbol{x}^{i,1})$.*

uncertainty *There exists an input $x^?_i$ such that for all subsets $C \subset \{\mathsf{P}_1, \ldots, \mathsf{P}_n\} \setminus \{\mathsf{P}_i\}$ with $|C| = t$ and $D = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\} \setminus (\{\mathsf{P}_i\} \cup C)$ and all inputs $\boldsymbol{x}$ for f there exists $x_D = \{x'_j\}_{j \in D}$ such that $f(\boldsymbol{x}[(i, x^?_i)]) = f(\boldsymbol{x}[x_D])$.*

We say that f is t-difficult *if f is t-difficult for all P_i.*

Intuitively, if a party has influence, then the function – at least sometimes – depends on the input of that party. If a party P_i has uncertainty, it means that for some input, called $x_i^?$, of P_i, if subset C is corrupt, they will not be able to figure out which input P_i has, no matter what the other inputs were: we can switch P_i's input to anything else and compensate for this by changing the inputs of the other honest parties such that the output is the same. One may think, for instance of the AND function: if P_i has input 0, the output is 0, but the adversary cannot know if this is because P_i or another honest party has a 0.

As examples of t-difficult functions consider the functions where each party has as input a bit and where the output is the AND or the XOR of these n bits. Other examples are general threshold functions, which output 1 iff at least some $0 < t' < n$ parties have input 1.

For a run of a protocol π and a given party P_i and a given point in the protocol we keep track of a set N_i which can be thought of as the parties that P_i has exchanged messages with, but it is defined with a slight twist. From the beginning we set all $N_i = \emptyset$. Whenever P_i sends a message, we update N_i to be the set of parties P_i has sent a message to or received a message from so far in the protocol. The definition is important so let us elaborate:

1. The set N_i is not updated at the time a message is received.
2. The set N_i *is* updated at the time a message is sent.
3. When N_i is updated we add all the messages that were received since the last time is was updated and we also add the outgoing message that triggered the update.

We say that a protocol has *t-floating input* for P_i if at each point in the protocol where $|N_i| \leq t$ it holds that P_i still did not read its input x_i. More formally, if we model P_i as an interactive Turing machine, it means that P_i did not access its input tape. We say that π has t-floating input if it has t-floating input for all parties.

For any run of a protocol we define a *revelation message* to be the message (if it exists) where before the message is sent it holds for at least one P_i that $|N_i| \leq t$ and after the message is received it holds for all P_i that $|N_i| \geq t + 1$. Notice that this implies that it is the size of the set N_i of the *sender* of the revelation message that crosses the threshold t, as N_i is not updated in response to receiving a message.

The *communication pattern* of an execution $\pi(\boldsymbol{x}; R)$ with input vector $\boldsymbol{x}$ and random tape vector R is the transcript seen by the adversary when no parties are corrupted, i.e., who sent a message to whom at which time and the length of those messages, but no contents of the messages and no input or output of any party. We assume that a communication pattern is encoded as a bit string. Let $Q : \{0,1\}^* \to \{0,1\}^*$ be a function on communication patterns. We use $Q(\pi(\boldsymbol{x}))$ to denote the random variable obtained by running π on the input distribution $\boldsymbol{x}$ and uniformly random R and applying Q to the resulting communication pattern and then outputting the output of Q.

Our proof strategy can be summarized as follows: we will first show that a protocol with floating inputs must have a revelation message, and that furthermore, $n-1$ players must receive a message after the revelation message was sent, with probability at least 1/2. This is quite straightforward and implies that floating input protocols must satisfy our lower bound. The second step is to show that any secure protocol for a difficult function f can be converted to a floating input protocol with the same message complexity. This is the most complicated part and uses in an essential way that the function is difficult and the assumption in our model that the number of parties does not grow with the security parameter.

Lemma 1 (input-independent communication pattern). *If π securely implements f with statistical security for t semi-honest corruptions for some $t \geq 0$, then it holds for any two input distributions $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ and all functions Q on communication patterns that $Q(\pi(\boldsymbol{x}_0))$ and $Q(\pi(\boldsymbol{x}_1))$ are statistically indistinguishable.*

Proof. This follows from the fact that when no parties are corrupted, the adversary still sees the communication pattern of $\pi(\boldsymbol{x}_0)$ and $\pi(\boldsymbol{x}_1)$ and hence can compute and output $Q(\pi(\boldsymbol{x}_0))$ respectively $Q(\pi(\boldsymbol{x}_1))$. However, when no parties are corrupted the simulator has the same view when $\boldsymbol{x}_0$ or $\boldsymbol{x}_1$ is used. The claim then follows from security against 0 semi-honest corruptions. □

Corollary 1 (input-independent communication complexity). *If π securely implements f with statistical security for t semi-honest corruptions for some $t \geq 0$, then it holds for any two input distributions $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ that $\mathsf{Msg}(\boldsymbol{x}_0)$ and $\mathsf{Msg}(\boldsymbol{x}_1)$ are statistically indistinguishable. Here, $\mathsf{Msg}(\boldsymbol{x})$ is the random variable that selects an input according to $\boldsymbol{x}$, runs the protocol and outputs the number of non-empty messages sent.*

Proof. Consider the function on communication patterns outputting the number of non-empty messages sent and then apply Lemma 1. □

Lemma 2 (revelation message). *If π has t-floating input and securely implements f with statistical security for t semi-honest corruptions and f is t-difficult, then it holds for all input distributions $\boldsymbol{x}$ that π has a t-revelation message except with negligible probability.*

Proof. If π does not have a t-revelation message for input distribution $\boldsymbol{x}$, then there exist a party P_i such that with non-negligible probability P_i exchanges messages with at most t parties in $\pi(\boldsymbol{x})$. From Lemma 1 it then follows that it holds for the input distributions $\boldsymbol{x}^{i,0}$ and $\boldsymbol{x}^{i,1}$ from the definition of f being t-difficult that with non-negligible probability P_i exchanges messages with at most t parties in $\pi(\boldsymbol{x}^{i,0})$ and also in $\pi(\boldsymbol{x}^{i,1})$. But since π has t-floating inputs, this implies that with non-negligible probability $\pi(\boldsymbol{x}^{i,0}) = \pi(\boldsymbol{x}^{i,1})$ as the output cannot depend on the input of P_i when P_i did not read its input, and all other parties have the same inputs in $\boldsymbol{x}^{i,0}$ and $\boldsymbol{x}^{i,1}$. However, by assumption $f(\boldsymbol{x}^{i,0}) \neq f(\boldsymbol{x}^{i,1})$ and we have a contradiction with correctness of π. □

Lemma 3 (another message after revelation message). *If π has t-floating input and securely implements f with statistical security for t semi-honest corruptions and f is t-difficult, then it holds for all input distributions and all pairs of distinct parties P_j and P_k that in a random run of $\pi(\boldsymbol{x})$ it holds except with negligible probability that when P_k is the sender of the revelation message, then the probability that P_j receives another message after P_k sent the revelation message is at least $\frac{1}{2}$.*

Proof. Assume for the sake of contradiction that there exist $\boldsymbol{x}$ and P_k and $\mathsf{P}_j \neq \mathsf{P}_k$ such that it happens with non-negligible probability that P_k is the sender of the revelation message and that when this happens P_j will receive another message after the revelation message is sent (but not received) with probability at most $\frac{1}{2} - c$, where c is non-negligible. It is a predicate of the communication pattern whether P_k sends the revelation message. It is also a predicate of the communication pattern whether P_j receives another message after the revelation message. Therefore it follows from Lemma 1 that it holds for any input distribution $\boldsymbol{x}$ with non-negligible probability that P_k is the sender of the revelation message and that when this happens then P_j will receive another message after the revelation message is sent (but not received) with probability at most $\frac{1}{2} - c'$, where $c' = c - \mathrm{negl}$ is non-negligible as c is non-negligible.

Consider now the particular input distribution which is $\boldsymbol{x}^{k,b}$ for a uniformly random bit b, where $\boldsymbol{x}^{k,0}, \boldsymbol{x}^{k,1}$ are the input vectors guaranteed by the definition of f being t-difficult (P_k has influence). In this case the output of all parties allow to determine the bit b, except with negligible probability. Assume without loss of generality that $f(\boldsymbol{x}^{i,b}) = b$. Let y be the distribution of the output of P_j in a random run on $\boldsymbol{x}^{i,b}$ conditioned on P_j not receiving another message after the revelation message. Notice that y can be sampled by P_j at the time right before the revelation message is sent, by simply assuming that no more messages will be received by P_j. However, at the point before the revelation message is sent P_k did not read its input x_k yet in the protocol, so y is perfectly independent of b. From this it follows that $\Pr[y = 0 \,|\, b = 0] = \Pr[y = 0 \,|\, b = 1] = 1 - \Pr[y = 1 \,|\, b = 1]$, so either $\Pr[y = 0 \,|\, b = 0] \leq \frac{1}{2}$ or $\Pr[y = 1 \,|\, b = 1] \leq \frac{1}{2}$. Assume that $\Pr[y = 0 \,|\, b = 0] \leq \frac{1}{2}$. Since $b = 0$ with probability $\frac{1}{2}$ and P_j receives another message with probability $\frac{1}{2} - c$ it happens with non-negligible probability that $b = 0$ and at the same time P_j does not receive another message and hence outputs according to distribution y, which implies that it happens with non-negligible probability that P_j does not output b, contradicting the correctness of the protocol. If we assume that $\Pr[y = 1 \,|\, b = 1] \leq \frac{1}{2}$, then a violation of correctness is reached using a symmetric argument. This concludes the proof. □

Lemma 4 (floating input). *Let f be a t-difficult n-party function and assume that π is an n-party protocol securely implementing f with statistical correctness and statistical privacy against t semi-honest corruptions. Then there exists a protocol π' with t-floating input which has the same security and is such that for any input distribution, the resulting communication patterns of π and π' are identically distributed.*

Proof. We prove the lemma by constructing π' from π. We prove the lemma for the weaker case where we construct π' where only P_1 has t-floating input. We can then obtain the general case by symmetry and hybrid arguments.

All parties in π' run as in π except P_1 who runs as follows. Initially, run as in π but with input $\beta_1 = x_1^?$ and a uniformly random tape ρ_1. Here, $x_1^?$ is the input value that a exists since f is t-difficult (the uncertainty condition for P_1). If about to send a message which would result in $|N_1| \geq t+1$, then first apply the following *input patching* procedure: Read the input x_1 and replace ρ_1 with a new random tape r_1 consistent with input x_1 and the communication so far. Specifically, sample r_1 using rejection sample as follows. Sample r_1 uniformly at random. Let T be the list of messages sent and received by P_1 so far, including who the message was exchanged with and in which round. Run the code of P_1 from π with input x_1 and random tape r_1 and feed P_1 the incoming message from T in the round in which they occurred. If this makes P_1 send the same messages as in T to the same parties and in the same rounds, then accept r_1, otherwise try again. Use $r_1 = \perp$ to denote that no acceptable r_1 exists. We now prove that if π is secure, then π' is secure.

We will actually prove something stronger, which implies that the correctness and the distribution of the communication pattern is also maintained. Namely we will prove that for all input distributions $\boldsymbol{x}$ it holds that the following distributions D_0 and D_1 are statistically indistinguishable: D_0 is obtained by sampling a random run of π on a random input sampled from $\boldsymbol{x}$ and then outputting $((x_1, r_1), (x_2, r_2), \ldots, (x_n, r_n))$, where x_i is the input of P_i and r_i is the random tape used by P_i. D_1 is obtained by sampling a random run of π' on a random input sampled from $\boldsymbol{x}$ and then outputting $((x_1, r_1), (x_2, r_2), \ldots, (x_n, r_n))$, where for $i = 2, \ldots, n$ the value x_i is the input of P_i and r_i is the random tape used by P_i and where x_1 is the input of P_1 and r_1 is the random tape sampled in the input patching procedure. From this it clearly follows that if π is correct, then π' is correct and it follows for all $t' \leq n$ that if π is secure against t' corruptions then π' is also secure against t' corruptions. Notice that to prove the claim for all distributions on $\boldsymbol{x}$ it is sufficient to prove that it holds for all fixed input vectors $\boldsymbol{x}$, so in the following we assume that $\boldsymbol{x}$ is a fixed value.

Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$. If $x_1 = x_i^?$, then the input patching procedure simply resamples r_1 with the same distribution as β_1 and hence D_0 and D_1 are identical. So, assume that $x_1 \neq x_1^?$ and that D_0 and D_1 are not statistically close. We show how to use this to break the t-security of π. Let $\boldsymbol{x}_0 = (x_1^?, x_2, \ldots, x_n)$ and $\boldsymbol{x}_1 = (x_1, x_2, \ldots, x_n) = \boldsymbol{x}$. We break the analysis into two cases. In case I we assume that $f(x_1?, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_n)$. To avoid confusion, note that the proof of case II in fact implies the result for case I. However, it is instructive to first see the proof of case I as a mental warm-up.

In case I we will run π on $\boldsymbol{x}_0$ or $\boldsymbol{x}_1$ and show how to distinguish with non-negligible advantage by corrupting just t parties which do not include P_1. This clearly demonstrates that π is not t-secure, as these t parties have the same inputs and outputs in $f(\boldsymbol{x}_0)$ and $f(\boldsymbol{x}_1)$ as only the input of P_1 differs and because $f(\boldsymbol{x}_0) = f(\boldsymbol{x}_1)$. So, assume that we attack a run of $\boldsymbol{x}_b$ for uniformly random b.

The adversary will observe the communication pattern of the protocol. Consider the point where P_1 sends a message that would make $|N_1| > t$ for the first time, and note that P_1 has communicated with at most t parties up to now, call this set of parties C. At this point the adversary corrupts the players in C[5]. Note that all messages sent by P_1 so far was sent to one of these parties. Use D to denote the set of parties which is not in $\{\mathsf{P}_1\} \cup C$. Now use rejection sampling to sample a random tape r_1 consistent with the communication between P_1 and the parties in C and input x_i to P_1. Note that if $b = 1$, this samples a string having the same distribution as the random tape used by P_1 in the protocol $\pi(\boldsymbol{x})$. If $b = 0$, then it samples a string having the same distribution as the random tape r_1 sampled by the input fixing procedure in $\pi'(\boldsymbol{x})$. Note that the parties in D have not communicated with P_1, so all the communication leaving the group D is with C. This means that the adversary knows all message going in or out of the group D. It can therefore use rejection sampling to sample a set of uniformly random tapes $\{(j, r_j)\}_{j \in D}$ for the parties in D consistent with the communication between C sand D and P_j for $j \in D$ having input x_j (where x_j is taken from $\boldsymbol{x}$). This perfectly reconstructs the distribution of the state of the parties in D. Then output $((x_1, r_1), (x_2, r_2), \ldots, (x_n, r_n))$. If $b = 0$, this is exactly D_0 and if $b = 1$ it is exactly D_1. But we assumed that D_0 and D_1 are not statistically close so we arrive at a contradiction with t-security of π: since the output is the same in the two cases, a simulator would see no difference between $b = 0$ and $b = 1$.

That brings us to case II. In this case, we can prove as above that if D_0 and D_1 can be distinguished, then we can also distinguish between $\pi(\boldsymbol{x}_0)$ and $\pi(\boldsymbol{x}_1)$ by just corrupting t parties at a point where $|N_1| \leq t$. The challenge is that $f(\boldsymbol{x}_0) \neq f(\boldsymbol{x}_1)$, so it does not follow easily from the definition of security that an adversary should *not* be able to distinguish with just this information. We now argue this in a more indirect way.

Consider the following experiment, which is parameterized by an (infinitely powerful) adversary A that outputs one bit:

1. Sample b uniformly at random.
2. Run $\pi(\boldsymbol{x}_b)$ until the point where it is about to happen that $|N_1| > t$. If this point does not occur, then perform the following at the end of the execution of the protocol.
3. Let C be the set of at most t parties defined as in case I above. A corrupts the parties in C. Let V be the joint view of these parties (their inputs, random tapes and messages received). Output $A(C, V)$ (here we abuse notation slightly by using A to denote both the adversary and the (arbitrary) function it calculates on the views).

We now claim that for any A, $\Pr[A(C, V) = b] - \frac{1}{2}$ is negligible. This will imply what we want: Note that one possible choice of A is as follows: use rejection sampling to produce a sample of D_b, exactly as we described in case I above.

[5] when we get to the actual proof in case II, we will construct a static adversary that always corrupts the same set.

Then output the best guess at whether the sample came from D_0 or from D_1. Since the claim holds for this particular A, D_0 and D_1 are statistically close.

So assume for the sake of contradiction that there exists A such that $\Pr[A(C,V) = b] - \frac{1}{2}$ is non-negligible.

Note that C may not be the same set in all runs of the protocol. Considering C as a random variable, we have that

$$\begin{aligned}
&\Pr[A(C,V) = b] - \frac{1}{2} \\
&= \sum_{C'} \Pr[C = C'] \Pr[A(C,V) = b | C = C'] - \left(\sum_{C'} \Pr[C = C'] \right) \frac{1}{2} \\
&= \sum_{C'} \Pr[C = C'] \left(\Pr[A(C,V) = b | C = C'] - \frac{1}{2} \right) .
\end{aligned}$$

Since the number of subsets of the parties is constant as a function of the security parameter, it now follows that we can find a fixed set C' of size at most t such that $\Pr[C = C']$ is non-negligible and such that $\Pr[A(C',V) = b \,|\, C = C'] - \frac{1}{2}$ is non-negligible.

We can then construct a new adversary A' which always corrupts C' and still guesses b with non-negligible advantage: If the set C actually occurring in the protocol equals C', it outputs $A(C',V)$, otherwise it outputs a uniformly random bit. Note that A' makes its guess at a point in time where $N_1 \subseteq C$. A''s advantage is non-negligible because $\Pr[A'(C',V) = b \,|\, C \neq C'] - \frac{1}{2}$ is negligible — we can only claim negligible here and not 0 as there might be a difference between $\Pr[C \neq C' \,|\, b = 0]$ and $\Pr[C \neq C' \,|\, b = 1]$. This difference, however, is negligible by Lemma 1.

We now want to show that such A' does not exist. To avoid ugly notation in the following, we will now use C to denote the set that A' always corrupts.

We start with some notation. Let D be the set of parties not in $C \cup \{\mathsf{P}_1\}$. Let $x_1^0 = x_1^?$ and $x_1^1 = x_1$ let x_D^1 be the inputs of the parties in D in $\boldsymbol{x}$. Let x_D^0 be the inputs x_D for the parties in D given by the definition of P_1 having uncertainty. We therefore have have by definition that $f(x_1^0, x_C, x_D^1) = f(x_1^1, x_C, x_D^0)$.

In this notation we have that $\boldsymbol{x}_0 = (x_1^0, x_C, x_D^1)$ and $\boldsymbol{x}_1 = \boldsymbol{x} = (x_1^1, x_C, x_D^1)$. Therefore our job is to prove that A' cannot distinguish $\pi(x_1^1, x_C, x_D^1)$ from $\pi(x_1^0, x_C, x_D^1)$. In the following, for a subset S of the parties, we use $[b,d]_S$ to denote the view of the parties S in an execution of $\pi(x_1^b, x_C, x_D^d)$. To complete the proof we have to show that at any point in the protocol where $N_1 \subseteq C$ it holds that $[0,1]_C \approx [1,1]_C$.

For a subset S of the parties, let $[b,d]_S^c$ denote the distribution of their views, conditioned on the parties in S having received at most c messages.

Obviously $[0,1]_C^0 \approx [1,1]_C^0$, since before C communicated with any party the view of players in C is just their own inputs and random tapes. We now prove by induction that $[0,1]_C^c \approx [1,1]_C^c$ for all constants c, as long as $N_1 \subseteq C$. The latter condition is extremely important because it implies that in all cases we consider, there is no communication between P_1 and D.

We assume that $[0,1]_C^c \approx [1,1]_C^c$ and prove that $[0,1]_C^{c+1} \approx [1,1]_C^{c+1}$. From the communication pattern being known by the adversary and being indistinguishable in $[0,1]_C^c$ and $[1,1]_C^c$ by Lemma 1 we can assume that we know which party P_j sends a message to C in round $c+1$.

Assume first that $\mathsf{P}_j \neq \mathsf{P}_1$. Let R_D be the procedure which gets input $[b,1]_C^c$, and then from the view of the communication between C and D in $[b,1]_C$ samples a joint state of all parties in D consistent with inputs x_D^1 and that communication and appends this state to $[b,1]_C$. We have that $R_D([b,1]_C^c) = [b,1]_{C,D}^c$ by construction and it follows from the induction hypothesis $[0,1]_C^c \approx [1,1]_C^c$ that $R_D([0,1]_C^c) \approx R_D([1,1]_C^c)$. So we conclude that in this case (where $\mathsf{P}_j \in D$) $[0,1]_{C,D}^c \approx [1,1]_{C,D}^c$. Put another way, given the state of C one can perfectly simulate the state of the parties in D since one knows their inputs and all communication going in and out of D. From the state of the parties in D in $[b,1]_{C,D}^c$ one can then sample a random run consistent with P_j being the next party to send a message to a party in C. This gives a sample from $[b,1]_{C,D}^{c+1}$. Since computation (in this case of the next message function) maintains statistical indistinguishability it follows from $[0,1]_{C,D}^c \approx [1,1]_{C,D}^c$ that $[0,1]_{C,D}^{c+1} \approx [1,1]_{C,D}^{c+1}$. It clearly follows from $[0,1]_{C,D}^{c+1} \approx [1,1]_{C,D}^{c+1}$ that $[0,1]_C^{c+1} \approx [1,1]_C^{c+1}$.

Assume then that $\mathsf{P}_j = \mathsf{P}_1$. Again, by induction hypothesis we have $[0,1]_C^c \approx [1,1]_C^c$. It follows from the security of the protocol that $[0,1]_C \approx [1,0]_C$ as the inputs and outputs of the parties in C are the same in the two executions considered and $|C| \leq t$. So in particular we have $[0,1]_C^c \approx [1,0]_C^c$. So we conclude by transitivity that $[1,0]_C^c \approx [1,1]_C^c$.

Since the next message comes from P_1 we can argue $[1,0]_C^{c+1} \approx [1,1]_C^{c+1}$ as we did for the above case, by sampling the state of P_1 from its known input and communication. As we noticed above we have $[0,1]_C \approx [1,0]_C$ and therefore in particular $[0,1]_C^{c+1} \approx [1,0]_C^{c+1}$. Combining these two we get $[0,1]_C^{c+1} \approx [1,1]_C^{c+1}$ as desired. □

Theorem 1. *Let π be the n-party function which securely implements a function f which is t-difficult, with statistical correctness and statistical privacy against t semi-honest corruptions. Then*

$$\mathsf{Msg}_{\texttt{lib}}(\pi) \geq \lceil (n(t+1)-1)/2 \rceil + n/2 - \frac{1}{2}$$

and

$$\mathsf{Msg}_{\texttt{con}}(\pi) \geq \lceil (n(t+1)-1)/2 \rceil + n - 1 \ .$$

Proof. We start by proving the bound for liberal communication complexity. By Lemma 4 we can assume that π has t-floating inputs. From Lemma 2 we then get that π has a revelation message for all input distributions, except with negligible probability. We now want to count the number of send and receive operations that have been executed just before the revelation message is sent. Since $|N_j| \geq t+1$ for all P_j *after* the revelation message is sent, it follows that after it is sent

$$\sum_{i=1}^{n} |N_i| \geq n(t+1) \ .$$

Notice that in this sum the revelation message is counted only once, but all other messages might be counted twice. Hence at least $(n(t+1)-1)/2+1$ messages were sent after the revelation message was sent. Therefore at least $(n(t+1)-1)/2$ messages were sent before the revelation message was sent. Since the number of messages sent is an integer, it follows that at least $\lceil (n(t+1)-1)/2 \rceil$ messages were sent. By Lemma 3, after the point where the revelation message is sent by some P_k each other party receives at least one more message with probability at least $\frac{1}{2} - \mathrm{negl}$. By linearity of expectation, this gives at least an expected $(n-1)(\frac{1}{2} - \mathrm{negl}(s))$ more messages. Since n is a constant in s we have that $n\,\mathrm{negl}(s) = \mathrm{negl}(s)$, so $\lim_{s\to\infty}(n-1)(\frac{1}{2} - \mathrm{negl}(s)) = (n-1)\frac{1}{2} = n/2 - \frac{1}{2}$. It is easy to see that for conservative message complexity we get to add $n-1$ instead of $(n-1)/2$: when we consider conservative message complexity, receiving a message with probability $\frac{1}{2}$ counts as 1 towards the message complexity. $\square$

We say that a function f is *t-very difficult* if it is t-difficult and in addition for P_i there exists P_j such that P_i and P_j has an embedded AND in the following sense: There exists an input vector $\boldsymbol{x}$ and inputs x_i^1 and x_i^0 for P_i and inputs x_j^1 and x_j^0 for P_j such that if we set $y_{b,c} = f(\boldsymbol{x}[(i, x_i^b), (j, x_j^c)]$ for $b, c \in \{0,1\}$, then $y_{0,0} \neq y_{1,1}$ and $y_{0,0} = y_{0,1} = y_{1,0}$. We note that the notion of an embedded AND (or, equivalently, an embedded OR) has been extensively studied in other settings, see [KKMO00] and references therein.) If f is t-very difficult we can improve the lower bound by $\frac{1}{2}$ message.

Theorem 2. *Let π be the n-party function which securely implements a function f which is t-very difficult, with statistical correctness and statistical privacy against t semi-honest corruptions. Then*

$$\mathsf{Msg}_{\mathtt{lib}}(\pi) \geq \lceil (n(t+1)-1)/2 \rceil + n/2$$

and

$$\mathsf{Msg}_{\mathtt{con}}(\pi) \geq \lceil (n(t+1)-1)/2 \rceil + n \ .$$

Proof (sketch). We start by proving the bound for liberal communication complexity. The proof follows the lines of the proof of Theorem 1, so we will only give a sketch. The extra $\frac{1}{2}$ message comes from the fact that we can now argue that even the sender of the revelation message must receive another bit of information after sending the revelation message and therefore must receive another message with probability at least $\frac{1}{2}$. To see this, note that if this was not the case, then it holds for all input distributions, by Lemma 1. Let P_k be the sender of the revelation message and let P_j be the party with which P_k has an embedded AND. Denote an execution of $\pi(\boldsymbol{x}[(j, x_j^b), (j, x_k^c)])$ by $[b, c]$. Assume that P_k receives a message after sending the revelation message with probability less than $\frac{1}{2}$.

In $[b, 0]$ it holds that the view of P_k is independent of b even at the end of the execution as the output and input of P_k are the same in the two executions. That implies that until P_k sends the revelation message it also holds in $[b, 1]$ that the view of P_k is independent of b, as $[b, 0]$ and $[b, 1]$ are perfectly indistinguishable to P_k until P_k actually reads its input. From this it follows that it also holds in $[b, 1]$ that the view of P_k is independent of b *after sending the revelation message*, as reading the input $x_j^c = 1$ cannot change that dependence on b as 1 is a constant and in particular independent of b and the view of P_k so far. But in $[b, 1]$ the output of P_k must be b by the correctness of π. Going from a situation where the view of P_k is independent of b to learning b requires that P_k receives a message with probability at least $\frac{1}{2}$. When we consider conservative message complexity, receiving a message with probability $\frac{1}{2}$ counts as 1 towards the message complexity. $\square$

Lower bounds for Perfect Security and Adaptive Corruption. Our model assume that the number of parties is constant as a function of the security parameter. The only place in our lower bound proofs where we used this assumption is in the proof of Lemma 4. If we consider perfect security, the proof simplifies greatly, and we can easily prove the lemma for any number of parties. Alternatively, if we consider adaptive security, note that the proof first constructs an adaptive adversary that breaks the protocol if our result is false and then converts it to an static adversary using the assumption on a constant number of parties. Therefore it is immediate that the lemma also holds for any number of parties and adaptive security. We conclude that all our lower bounds for this section hold for any number of parties, if we consider perfect or adaptive security.

4.2 Individual Round Complexity

Consider now an n-player protocol π that is executed on a synchronous network. We can define a (possibly empty) set M_π of players with *minimal interaction*, consisting of players whose only communication is to each send a message to a subset of the parties not in M_π and then later, after all parties in M_π have sent all their messages, each receive a message from a subset of the parties not in M_π.

Theorem 3. *Assume $n = 2t + 1$ parties, where each party P_i holds input bit b_i. A protocol π that computes $b_1 \wedge \cdots \wedge b_n$ with perfect correctness and statistical privacy against t semi-honest corruptions must have $|M_\pi| \leq t$.*

Proof. Assume for contradiction that M_π has size $t + 1$. Then we can construct from π a 3-party protocol for players A, B and C, where player A emulates the t players not in M_π, B emulates t of the players in M_π, and C emulates the last player in M_π. Each party will have a single bit as input and will use that bit as input to each of the parties it is emulating. If π is secure, then clearly the 3-party protocol securely computes the AND of the inputs from the 3 players, provided at most 1 is passively corrupt, as corrupting any of A, B and C will corrupt at most t emulated parties. Moreover, the 3 party protocol will have only 4 messages. Namely, the one party from M_π emulated by C will send one message to A and later receive exactly one message from A, as A emulated exactly the parties not in M_π. The same is true for all the emulated players in

B, they will all send exactly one message to a player in A and receive back one message from a player in A. Furthermore, since they all send their messages to the players in A before they received any messages from A, we can let B send all the messages as one message. In the same way we can let A return all the messages as one message. Since there is no communication between parties in M_π, there is no communication between B and C. Hence all other communication takes place inside A. However, communicating just 4 message is in contradiction to Theorem 2, which says that 6 messages are required.

Theorem 4. *Assume $n = 3t + 1$ parties, where each party P_i holds input bit b_i. A protocol π that computes $b_1 \wedge \cdots \wedge b_n$ with statistical correctness and statistical privacy against t malicious corruptions must have $|M_\pi| \leq t$.*

Proof. If we assume a contradiction we can as above reduce it to the case with $n = 4$ and $t = 1$. We let A simulate t parties with optimal communication complexity. We let B simulate the last party with optimal communication complexity. We let C and D each simulate t of the remaining parties. We set the input of D to be 1 and we denote the inputs of A, B and C by a, b and c. The communication pattern is as follows. First A sends two messages to C and D. Denote the message sent to C by g. At the same time B sends two messages to C and D. Denote the message sent to C by h. By privacy against a semi-honest corruption of C we know that g is independent of a. Clearly the message h is independent of a. Furthermore, since g and h were computed by two different parties which did not communicate before sending these messages, and the parties do not have a source of correlated randomness, g and h are independent. It follows that (g, h) is independent of a. However, by security of one malicious corruption the protocol should still terminate with the correct result if at this point D stops participating in which case C receives no further information. Clearly C cannot always compute the correct result with good probability when its view is independent of a. $\square$

5 Upper Bounds

In this section we give four constructive upper bounds, one for individual round complexity of secure function evaluation in the face of semi-honest corruptions, then one for individual round complexity of broadcast in the face of malicious corruptions, one for individual round complexity of secure functional evaluation in the face of malicious corruptions, and finally one for message complexity in the face of semi-honest corruptions.

5.1 Individual Round Complexity, Semi-honest Security

We first give a construction with minimal individual round complexity for a group of $t < n/2$ parties in the face of semi-honest corruption.

Theorem 5. *For every poly-time n-party function f, there exists a poly-time function evaluation protocol computing f between $n = 2t + 1$ parties with perfect correctness and perfect privacy against t semi-honest corruptions, where t parties have round complexity two. Specifically, these t parties first in parallel each send one message to the $n - t$ other parties and then later each receives one message from the same $n - t$ parties.*

Proof. We design a protocol where it is the parties $I = \{\mathsf{P}_{n-t+1}, \ldots, \mathsf{P}_n\}$ which have round complexity two. We denote each of the t parties in I generically by P_i and we denote the parties in $J = \{\mathsf{P}_1, \ldots, \mathsf{P}_{n-t}\}$ generically by P_j.

Use D to denote the pre-processing distribution of a secure function evaluation protocol for the pre-processing model with $n' = t + 1$ parties and up to t semi-honest corruptions. Let $(D, \pi_{\mathtt{pre-pro}})$ be a protocol for this model with perfect correctness and perfect privacy for t semi-honest corruptions.

Let $\pi_{\mathtt{hon-maj}}$ be a secure function evaluation protocol for the function f for a model with $n = 2t + 1$ parties and assume that it has perfect correctness and perfect privacy against t semi-honest corruptions. Assume that $\pi_{\mathtt{hon-maj}}$ has round complexity ℓ. We can assume that $\pi_{\mathtt{hon-maj}}$ runs as follows in round r: first each party sends one message to each other party which adds this message to its state. Then it applies a round function $R^{i,r}$ which computes the new state of party P_i. The initial state of a party is just its input x_i.

Our protocol π proceeds as follows. First each P_i will additively secret share its input x_i among the parties P_j, i.e., it samples uniformly random shares $x_{i,j}$ for which $x_i = x_{i,1} \oplus \cdots \oplus x_{i,n-t}$ and securely sends $x_{i,j}$ to P_j. At the same time it will for $r = 1, \ldots, \ell$ sample $(d_1^{i,r}, \ldots, d_{n-t}^{i,r}) \leftarrow D$ and send $d_j^{i,r}$ to P_j. Notice that at this point the initial state of each P_i is secret shared among the parties in J. We will keep the invariant that at each round in the protocol $\pi_{\mathtt{hon-maj}}$ the state of P_i in $\pi_{\mathtt{hon-maj}}$ is secret shared among the parties in J. Each round in $\pi_{\mathtt{hon-maj}}$ is emulated as follows.

1. If $\mathsf{P}_j \in J$ is to send a message m to $\mathsf{P}_k \in J$, then it sends m over the secure channel to P_k.
2. If $\mathsf{P}_j \in J$ is to send a message m to $\mathsf{P}_i \in I$, then it additively secret shares m among the parties J and this secret sharing is added to the secret shared state of P_i.
3. If $\mathsf{P}_i \in I$ is to send a message m to $\mathsf{P}_k \in I$, then m is by the invariant already additively secret shared among the parties J. The parties in J can therefore just add this secret sharing to the secret shared state of P_k.
4. If $\mathsf{P}_i \in I$ is to send a message m to $\mathsf{P}_j \in J$, then m is additively secret shared among the parties J as part of the secret shared state of P_i. The parties in J can therefore reconstruct this message towards P_j.
5. If $\mathsf{P}_j \in J$ is to apply the round function $R^{j,r}$, then it simply applies it to its state.
6. If $\mathsf{P}_i \in I$ is to apply the round function $R^{i,r}$, then the parties in J uses the the preprocessed values $(d_1^{i,r}, \ldots, d_{n-t}^{i,r})$ to do secure function evaluation of the augmented round function $\bar{R}^{i,r}$ which reconstructs the state of P_i from

the secret sharing of the state held by the parties in J, then applies $R^{i,r}$ and outputs an additive secret sharing of the new state.

After all ℓ rounds of $\pi_{\mathtt{hon-maj}}$ have been emulated, the secret-shared state of P_i contains its output y_i. The parties in J reconstructs this y_i towards P_i. At this point all n parties received their outputs.

It should be clear that this protocol has perfect correctness, as $\pi_{\mathtt{pre-pro}}$ and $\pi_{\mathtt{hon-maj}}$ both have perfect correctness.

As for perfect privacy, note that if at most t parties are corrupted, then the additive secret sharings among the t parties in J leaks no information, and can indeed be efficiently simulated by just giving all corrupted parties uniformly random shares.

Furthermore, if $\mathsf{P}_i \in I$ is honest, then the emulation of P_i in $\pi_{\mathtt{hon-maj}}$ is perfectly private, as P_i is perfectly acting as the trusted third party of the preprocessing model. We can in particular replace the emulation of P_i by an ideal function evaluation of the augmented round function.

Since the additive secret sharing of the inputs and outputs of the augmented round function can be efficiently simulated towards the t corrupted parties without knowing the inputs or outputs, we can replace the ideal evaluation of the augmented round function by an ideal evaluation of the actual round function on the actual state of P_i and then just simulate the secret sharing of the inputs and outputs using uniformly random shares. But having an ideal evaluation of the round function of an honest P_i is exactly the same as just having P_i participate in the protocol. So at this point we have arrived at the protocol $\pi_{\mathtt{hon-maj}}$. Since there are at most t corrupted parties we can then appeal to the security of $\pi_{\mathtt{hon-maj}}$.

Constructing an explicit simulator of π from the simulators of $\pi_{\mathtt{pre-pro}}$ and $\pi_{\mathtt{hon-maj}}$ along the lines of the above sketch is straight forward and we skip the technical details. $\square$

5.2 Individual Round Complexity, Broadcast

We now turn our attention to the individual round complexity of secure broadcast. Secure broadcast from P_i to the parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ is defined to be the secure function evaluation of the function $x_i = f(x_1, \ldots, x_n)$ in the face of malicious corruptions, i.e., P_i communicates x_i to all parties and it is guaranteed that all parties receive the same x_i even if P_i and/or some of the other parties are malicious. By secure broadcast we mean a protocol which allows any of the n parties to broadcast to all the other parties.

It is possible to implement broadcast securely against $t < n/3$ maliciously corrupted parties in a synchronous network with authenticated channels (note that secure channels are not needed for broadcast). It is furthermore possible to do so using a protocol where the honest parties are deterministic. See for instance [BDGK91].

The above protocol is for the setting with $t < n/3$ maliciously corrupted parties. We later need to do broadcast in a setting with $t < n/2$ maliciously corrupted parties. It is actually known that broadcast is impossible in such a setting. We can, however, implement broadcast if we assume $t < n/3$ for just the first round. To show this we need the following lemma.

Lemma 5. *Consider any protocol π for n parties which is perfectly correct and has statistical privacy against t maliciously corrupted parties computing a function f. Assume that $\mathsf{P}_{n-t+1}, \ldots, \mathsf{P}_n$ have no inputs, i.e., $f(x_1, \ldots, x_n) = g(x_1, \ldots, x_{n-t})$. Assume also that these parties are not to receive outputs. Assume furthermore that the protocol remains secure even if all messages sent and received by $\mathsf{P}_{n-t+1}, \ldots, \mathsf{P}_n$ are given to the adversary and assume that these parties are deterministic. Then there also exists a protocol π' which is statistically correct and has statistical privacy against t maliciously corrupted parties computing the function f in which $\mathsf{P}_{n-t+1}, \ldots, \mathsf{P}_n$ each sends a message to each of the parties $\mathsf{P}_1, \ldots, \mathsf{P}_{n-t}$ in the first round and then sends or receives no further messages.*

Proof. The parties $I = \{\mathsf{P}_1, \ldots, \mathsf{P}_{n-t}\}$ will simply emulate the parties $J = \{\mathsf{P}_{n-t+1}, \ldots, \mathsf{P}_n\}$. Each $\mathsf{P}_i \in I$ will run a copy of each $\mathsf{P}_j \in J$. Since P_j has no input, the parties P_i will agree on the initial states of all P_j. Whenever P_j wants to send a message, all P_i will know this message and the appropriate receiver will just take that message as if having been sent by P_j. If the receiver is a party $\mathsf{P}_j \in J$ all $\mathsf{P}_i \in I$ will input the message to their local copy of P_j. In each round all parties $\mathsf{P}_i \in I$ apply the deterministic round function of each P_j to their own local copy. This maintains agreement on the state of all the emulated P_j.

The only problematic case is when some $\mathsf{P}_i \in I$ wants to send a message m to some $\mathsf{P}_j \in J$. In that case P_i must send m to all parties in I such that they can input m to P_j. We have to ensure that P_i sends the same m to all parties in I, or they might end up with inconsistent versions of P_j. We ensure this by letting P_i broadcast the message m. The only problem is that we do not have a broadcast channel. We will therefore let P_j create one using pre-processing. This will be done using the one round of messages that P_j sends in the first round, as detailed now.

It is shown in [PW92] that there exists a protocol (P, π) for the pre-processing model which implements broadcast between n' parties secure against t malicious corruptions for any $t < n'$. We can therefore let each $\mathsf{P}_j \in J$ sample $(p_{j,1}, \ldots, p_{j,n'}) \leftarrow P$ and send $p_{j,i}$ securely to P_i. Whenever $\mathsf{P}_i \in I$ is to send m to all parties in I, the parties run π on the pre-processed values $(p_{j,1}, \ldots, p_{j,n'})$ and with P_i having input m. Note that each $\mathsf{P}_j \in J$ preprocessed his own broadcast channel. This is the broadcast channel that is to be used when message are sent *to* P_j in the emulated protocol. If P_j is honest, the pre-processing is computed as it should, and thus the broadcast protocol will indeed ensure that m is delivered consistently, and hence the emulated P_j will be run correctly and consistently by all honest parties in I. If P_j is corrupted, it might deliver incorrect pre-processed values. In that case the broadcast might not work correctly. In that case the parties in I might get inconsistent views of P_j and might therefore later see inconsistent values of what P_j is sending. This, however, is no worse than the emulated P_j being corrupted and this case only happens when the actual P_j is maliciously corrupted, so the emulated protocol can tolerate this. $\square$

If we plug the protocol from [BDGK91] into the above lemma we get this corollary.

Corollary 2. *There exists a protocol $\pi_{\texttt{broad}}$ for n parties which is statistically correct and which allows any party P_i (with $i \leq n-t$) to broadcast to the parties $\mathsf{P}_1, \ldots, \mathsf{P}_{n-t}$. It is secure against t malicious corruptions for $t < n/3$. The parties $\mathsf{P}_{n-t+1}, \ldots, \mathsf{P}_n$ each sends one message to each of the parties $\mathsf{P}_1, \ldots, \mathsf{P}_{n-t}$ in the first round and otherwise has no communication.*

5.3 Individual Round Complexity, Secure Function Evaluation

We now turn our attention to secure function evaluation in the face of malicious corruptions.

Theorem 6. *For every poly-time n-party function f, there exists a poly-time function evaluation protocol computing f between $n = 3t + 1$ parties with statistical correctness and statistical privacy against t maliciously corrupted parties, where t parties have round complexity two. Specifically, these t parties first each sends one message to the $n - t$ other parties in parallel and then later each receives one message from the same $n - t$ parties.*

Proof. As usual, $I = \{\mathsf{P}_1, \ldots, \mathsf{P}_{n-t}\}$ and $J = \{\mathsf{P}_{n-t+1}, \ldots, \mathsf{P}_n\}$. In [RB89] a statistically correct and statistically private protocol for secure function evaluation of any function g is given for the setting with n' parties of which at most $t < n'/2$ parties are maliciously corrupted. The protocol is for the setting with secure point-to-point channels plus a broadcast channel allowing any party to broadcast to the other n' parties. Denote this protocol by $\pi_{\texttt{RB}}$. Set $n' = n - t$. We are going to let the parties I run $\pi_{\texttt{RB}}$ to compute a particular function g derived from f. In doing that they will implement the broadcast channel using $\pi_{\texttt{broad}}$ from Corollary 2 with the parties in J providing the pre-processing.

We will use a robust secret sharing scheme $(\mathsf{sha}, \mathsf{rec})$ for n' parties and $t < n/2$ corruptions to let the parties in J provide inputs. Such a scheme is trivial to derive from, e.g., the verifiable secret sharing scheme constructed in [RB89], and has the following properties:

Privacy The joined distribution of any t positions from a random sample $(v_1, \ldots, v_{n'}) \leftarrow \mathsf{sha}(v)$ does not depend on the value v.

Robustness Sample $(v_1, \ldots, v_{n'}) \leftarrow \mathsf{sha}(v)$ for a value v chosen by the adversary. Now give t of the positions v_i to the adversary and let it replace them by v_i'. The positions are chosen by the adversary. For the remaining $n' - t$ positions, let $v_i' = v_i$. Then $\mathsf{rec}(v_1', \ldots, v_{n'}') = v$, except with probability 2^{-s}, where s is the statistical security parameter.

The function g takes $n-t$ inputs, $g(X_1, \ldots, X_{n-t})$, where each X_i is of the form $(x_i, x_{n-t+1,i}, \ldots, x_{n,i})$. It outputs

$$f(x_1, \ldots, x_{n-t}, \mathsf{rec}(x_{n-t+1,1}, \ldots, x_{n-t+1,n-t}), \ldots, \mathsf{rec}(x_{n,1}, \ldots, x_{n,n-t}))\ .$$

The overall protocol then runs as follows.

1. Each $\mathsf{P}_j \in J$ sends the pre-processing needed for $\pi_{\mathtt{broad}}$ to the parties in I and at the same time samples $(x_{j,1}, \ldots, x_{j,n-t}) \leftarrow \mathsf{sha}(x_j)$ and sends $x_{j,i}$ to $\mathsf{P}_i \in I$.
2. Each $\mathsf{P}_i \in I$ computes $X_i = (x_i, x_{n-t+1,i}, \ldots, x_{n,i})$.
3. The parties in I use the pre-processing provided in Step 1 to run $\pi_{\mathtt{broad}}$ and use the emulated broadcast channel to run $\pi_{\mathtt{RB}}(X_1, \ldots, X_{n-t})$.
4. When $\mathsf{P}_i \in I$ learns the output $y = \pi_{\mathtt{RB}}(X_1, \ldots, X_{n-t})$ it sends y to all parties in J.
5. Each party $\mathsf{P}_j \in J$ receives an output y_i from each $\mathsf{P}_i \in I$ and outputs the value y which occurs most often in the list $(y_1, \ldots, y_{n-t})$.

It follows directly from the security of $(\mathsf{sha}, \mathsf{rec})$, $\pi_{\mathtt{broad}}$ and $\pi_{\mathtt{RB}}$ that the protocol is private and that the honest parties in I learn the correct output y, except with negligible probability. Since there are $n' \geq 2t+1$ parties in I and at most t corrupted parties in I, it follows that there is a majority of honest parties in I. Hence, the honest parties in J will also learn the correct output y. $\square$

5.4 Message Complexity, Semi-Honest Security

We now turn our attention to the message complexity of secure function evaluation in the presence of semi-honest corruptions. We consider protocols with n parties which are perfectly secure against t semi-honest corruptions. We present an optimal construction for $t = 1$ for computing functions in PSM_{eff} as defined in Definition 1.

Theorem 7. *For every poly-time n-party function f in PSM_{eff}, there exists a poly-time function evaluation protocol π computing f between n parties with perfect correctness and perfect privacy against $t = 1$ semi-honest corruptions, for which* $\mathsf{Msg}_{\mathtt{lib}}(\pi) = (3n+1)/2$, $\mathsf{Msg}_{\mathtt{wor}}(\pi) \leq \lceil (3n+1)/2 \rceil$, *and* $\mathsf{Msg}_{\mathtt{con}}(\pi) = 2n$.

Proof. We first look at the restricted setting where P_n has no input and is the only player to learn the output, i.e., we look at secure function evaluation of $(\epsilon, \ldots, \epsilon, y) = f(x_1, \ldots, x_n)$, where ϵ is the empty string and $y = h(x_1, \ldots, x_{n-1})$ for an $(n-1)$-party function h.

Let $(R, M_1, \ldots, M_{n-1})$ be a PSM protocol for h and consider the following protocol π_1.

1. P_1 samples $r \leftarrow R$.
2. P_1 sends r to P_i for $i = 2, \ldots, n-1$.
3. For $i = 1, \ldots, n-1$, party P_i sends $m_i = M_i(x_i, r)$ to P_n.
4. P_n outputs $y = g(m_1, \ldots, m_{n-1})$.

Assume that P_n is corrupted. The view of P_n in the real world is

$$(M_1(x_1, r), \ldots, M_{n-1}(x_{n-1}, r))$$

for a random sample $r \leftarrow R$. The view of P_n in the ideal model is

$$y = f(x_1, \ldots, x_n) = h(x_1, \ldots, x_{n-1}) = g(M_1(x_1, r), \ldots, M_{n-1}(x_{n-1}, r)) \ .$$

Privacy then follows from the security of the PSM protocol.

Assume that $\mathsf{P}_i \neq \mathsf{P}_n$ is corrupted. The view of P_i in the real world is (x_i, r). The view of P_i in the ideal model is x_i. We can simulate the real world view from the ideal view simply by sampling $r \leftarrow R$ and then outputting (x_i, r).

We now extend the above protocol to a protocol π_2 which allows P_n to have an input and where all parties get the output, i.e., we look at secure function evaluation of $y = f(x_1, \ldots, x_n)$. We first present and analyze a simple solution and then later modify it slightly to reduce the number of messages sent. The simple solution is to let P_n additively secret share x_n as $x_n = s_1 \oplus s_2$ and send s_1 to P_1 and send s_2 to P_2. Then apply protocol π_1 to the function

$$h'((x_1, s_1), (x_2, s_2), x_3, \ldots, x_{n-1}) = f(x_1, \ldots, x_{n-1}, s_1 \oplus s_2)$$

and let P_n send the output to all the other parties. We can do this as h' clearly is in non-deterministic log-space if f is in non-deterministic log-space. Note that this simple protocol adds $n+1$ more message. Sending the output y to all parties is obviously secure as this value is also in the view of all parties in the ideal model. Only P_1, P_2 and P_n have any further extra values in the view. The extra values of P_n are s_1 and s_2 such that $x_n = s_1 \oplus s_2$. These are easy to simulate from the view of P_n in the ideal model which includes x_n: simply sample an additive secret sharing of x_n. The extra value of P_1 is s_1. This value is uniformly random and independent of x_n, so it can be simulated by just sampling it uniformly at random. Similarly for P_2.

Since s_1 is uniformly random and independent of x_n, we can save one message in the protocol by letting P_1 pick s_1 uniformly at random and send it to P_n along with the message that it already sends to P_n. The view of all parties will be the same in the modified protocol. The only difference is that the direction of one message was flipped. This gives the following secure protocol.

Let $(R, M_1, \ldots, M_{n-1})$ be a PSM protocol for the function h' described above.

1. P_1 samples $r \leftarrow R$.
2. P_1 sends $m_1 = M_1(x_1, r)$ to P_n along with a uniformly random share s_1.
3. P_n sends $s_2 = x_n \oplus s_1$ to P_2.
4. P_1 sends r to P_i for $i = 2, \ldots, n-1$.
5. For $i = 2, \ldots, n-1$, party P_i sends $m_i = M_i(x_i, r)$ to P_n.
6. P_n sends $y = g(m_1, \ldots, m_{n-1})$ to $\mathsf{P}_1, \ldots, \mathsf{P}_{n-1}$.

To further reduce the message complexity, we will now apply two additional message-reduction tricks. Using the first one we reduce the $2(n-2)$ messages in Steps 4 and 5 to just $n-1$ messages: Instead of having all parties send to P_n, we will let P_1 send his "PSM-contribution" to P_2, who appends his contribution and sends a message to P_3, etc. until P_n receives everything. In order to make sure that only P_n learns all the contributions, P_1 will send $n-1$ one-time pads to P_n and also pass them on to the other players who can use them to one-time pad encrypt their contributions.

With the second trick we reduce the number of messages in Step 6 from $n-1$ to $\lceil (n-1)/2 \rceil$. We let P_1 choose a random bit w which will be sent to all other players appended to the "PSM-contributions", thus not requiring additional message(s). Now, P_n can communicate the result, y, to the other players in the following way: if $y \oplus w = 0$ then P_n sends a bit 0 to players $P_1, \ldots, P_{\lceil (n-1)/2 \rceil}$ and does not send any message to the players $P_{\lceil (n-1)/2 \rceil + 1}, \ldots P_{n-1}$; otherwise (if $y \oplus w = 1$) then P_n sends a message 0 to the players $P_{\lceil (n-1)/2 \rceil + 1}, \ldots P_{n-1}$ and does not send any message to the players $P_1, \ldots, P_{\lceil (n-1)/2 \rceil}$. Observe that all players can retrieve the computed value y, and that the number of messages sent during that stage is at most $\lceil (n-1)/2 \rceil$. Both tricks can be implemented as follows. We replace steps 4, 5 and 6 by the following procedure:

1. P_1 samples uniformly random bit strings $p_2, \ldots, p_{n-1}$ where p_i has the same length as m_i. He also samples a uniformly distributed bit w. Then P_1 sends $(p_2, \ldots, p_{n-1}), w$ to P_n. This can be done in Step 2 above and therefore does not add another message.
2. P_1 sends $(r, p_2, \ldots, p_{n-1}), w$ to P_2.
3. Then for $i = 2, \ldots, n-1$ party P_i receives $(r, c_2, \ldots, c_{i-1}, p_i, p_{i+1}, \ldots, p_{n-1}), w$ from P_{i-1} and then sends $(r, c_2, \ldots, c_{i-1}, c_i, p_{i+1}, \ldots, p_{n-1}), w$ to P_{i+1}, where $c_i = M_i(x_i, r) \oplus p_i$, except that P_{n-1} does not send r to P_n.
4. Then P_n receives $(c_2, \ldots, c_{n-1})$ from P_{r-1} and for $i = 2, \ldots, n-1$ computes $m_i = c_i \oplus p_i$.
5. P_n computes the result y using the PSM protocol. Now, if $y \oplus w = 0$ then it sends 0 to all of players $P_1, \ldots, P_{\lceil (n-1)/2 \rceil}$ (and no message to the other players). Otherwise (if $f \oplus w = 1$) it sends 0 to all of players $P_{\lceil (n-1)/2 \rceil + 1}, \ldots P_{n-1}$ (and no message to the other players). Each P_i will observes if a message was received from P_n, and, using its index and w, computes y.

It is easy to see that this is perfectly correct. As for perfect security against one semi-honest corruption, consider the values c_i seen by P_j for $i < j < n$. Since P_j does not know p_i, c_i is a one-time pad encryption of m_i. All other values seen by a single party clearly leak no information on the input other than what is implied by y. For a given input, the average number of messages sent by P_n in Stage 5 is $(1/2)(\lceil (n-1)/2 \rceil + \lfloor (n-1)/2 \rfloor) = n/2 - 1/2$. (Whatever the value of w is, at most $\lceil (n-1)/2 \rceil \leq n/2$ messages are sent by P_n at Step 5.) The average number of messages sent by the protocol is therefore $n+1+n/2-1/2 = 3n/2+1/2$ (and in the worst case the number of message sent is $n+1+\lceil (n-1)/2 \rceil \leq 3n/2+1$, if n is even.) However, since all parties except P_n may potentially receive a message in the last step, the conservative message complexity is $2n$. $\square$

If we set $t = 1$ in our previous lower bound for liberal message complexity, we get $3n/2$, matching the upper bound of Theorem 7 except for $1/2$ a message. The conservative message complexity of the protocol in Theorem 7 is clearly $2n$ which matches the lower bound for conservative message complexity of 1-very difficult functions. So we have matching upper and lower bounds for the conservative message complexities of 1-very difficult functions in non-deterministic log space. We leave it as an open problem to find matching bounds for any $t > 1$.

Finally, we consider computing the XOR of one input bit from each player. This is the primary example of a function that is t-difficult but not t-very difficult. We can construct a protocol for this function, secure for $t = 1$ from the proof of Theorem 7: We observe that there is no need for P_n to secret share his input, instead we use the PSM protocol to let P_n learn $b_1 \oplus \cdots \oplus b_{n-1}$. This is secure because this value would anyway follow from the output and P_n's own input. P_n computes the output $b_1 \oplus \cdots \oplus b_n$ and sends it to the other players in the randomised fashion described in the protocol. The liberal and conservative complexties of this protocol are $3n/2 - 1/2$ and $2n - 1$, matching the lower bounds we showed for 1-difficult functions.

6 Acknowledgements

Work done in part while some of the the authors visited Simons Institute. First and second author acknowledge support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61061130540) for the Sino-Danish Center for the Theory of Interactive Computation, within which part of this work was performed; and also from the CFEM research center (supported by the Danish Strategic Research Council) within which part of this work was performed. The second author was partially supported by the European Research Council Starting Grant 279447, the second partially supported by the European Research Council Advanced Grant MPCPRO. The third author acknowledges partial support by NSF grants 09165174, 1065276, 1118126 and 1136174, US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. This material is also based upon work supported in part by DARPA Safeware program. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. Research by the fourth author partially supported by ANR project RDAM.

References

[BDGK91] Amotz Bar-Noy, Xiaotie Deng, Juan A. Garay, and Tiko Kameda. Optimal amortized distributed consensus (extended abstract). In Sam Toueg, Paul G. Spirakis, and Lefteris M. Kirousis, editors, *Distributed Algorithms, 5th International Workshop, WDAG '91, Delphi, Greece, October 7-9,* *1991, Proceedings*, volume 579 of *Lecture Notes in Computer Science*, pages 95–107. Springer, 1991.

[BGT13] Elette Boyle, Shafi Goldwasser, and Stefano Tessaro. Communication locality in secure multi-party computation. In *Theory of Cryptography*, pages 356–376. Springer, 2013.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10. ACM, 1988.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.

[CCD87] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (abstract). In Carl Pomerance, editor, *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, volume 293 of *Lecture Notes in Computer Science*, page 462. Springer, 1987.

[CCG$^+$15] Nishanth Chandran, Wutichai Chongchitmate, Juan A. Garay, Shafi Goldwasser, Rafail Ostrovsky, and Vassilis Zikas. The hidden graph model: Communication locality and optimal resiliency with adaptive faults. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 153–162, 2015.

[CK93] Benny Chor and Eyal Kushilevitz. A communication-privacy tradeoff for modular addition. *Information Processing Letters*, 45(1), 1993.

[DPP14] Deepesh Data, Manoj M Prabhakaran, and Vinod M Prabhakaran. On the communication complexity of secure computation. In *Advances in Cryptology–CRYPTO 2014*, pages 199–216. Springer, 2014.

[DZ13] Ivan Damgård and Sarah Zakarias. Constant-overhead secure computation of boolean circuits using preprocessing. In *TCC*, pages 621–641, 2013.

[FKN94] Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 554–563, 1994.

[GIPR] Mira Gonen, Yuval Ishai, Manoj Prabhabkahan, and Mike Rosulek. private communication. In *Unpublished work*.

[IK97] Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *ISTCS*, pages 174–184, 1997.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 294–304. IEEE, 2000.

[KKMO00] Joe Kilian, Eyal Kushilevitz, Silvio Micali, and Rafail Ostrovsky. Reducibility and completeness in private computations. *SIAM J. Comput.*, 29(4):1189–1208, 2000.

[PW92] Birgit Pfitzmann and Michael Waidner. Unconditional byzantine agreement for any number of faulty processors. In Alain Finkel and Matthias Jantzen, editors, *STACS 92, 9th Annual Symposium on Theoretical Aspects of Computer Science, Cachan, France, February 13-15, 1992, Proceedings*, volume 577 of *Lecture Notes in Computer Science*, pages 339–350. Springer, 1992.

[RB89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 73–85. ACM, 1989.