Unconditionally Secure Computation with Reduced Interaction

Ivan Damgård and Jesper Buus Nielsen[⋆]

Dept. of Computer Science, Aarhus University

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 non-trivial function (such as the AND of several input bits), assuming that all players have input and get output. We show that for n players and t corruptions, $n(t+3)/2$ messages is necessary, this holds already for semi-honest and static corruption. Note that for functions that can be securely computed in constant round, this bound is tight up to a constant factor. For the case $t = 1$ and semi-honest security, we show that $2n$ messages is also sufficient to compute a rich class of functions efficiently, showing that the bound is exact for $t = 1$.

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. Providing a positive answer seems to require completely new ideas for protocol design. 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, also this is shown to be optimal.

1 Introduction

In Multiparty Computation n players want to compute an agreed function on privately held inputs, such that the desired result is correctly computed and is

[⋆] The authors 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. Partially supported by the European Research Commission Starting Grant 279447.

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 round and efficiently in the circuit size of the function, but this is 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?

In this paper, we study this question with respect to two related but different measures of interaction: message complexity and round complexity, and we consider synchronous networks throughout.

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. In this way, we force protocols to have a fixed communication patterns.

However, using this measure to prove *lower* bounds is not satisfying. After all, it could be that one could do protocols with a very small 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. Put differently, we should worry that using conservative messages complexity makes it too easy to prove lower bounds. Therefore, we also define *liberal message complexity*, where the protocol is only charged for messages that are *explicitly* sent, and where we also consider the *expected* number of messages rather than the maximum. We discuss these measures in more detail later, when we define them formally.

Our results are as follows: We show that for n players and t semi-honest and static corruptions, $\lceil (n(t+3)-1)/2 \rceil$ messages is necessary to compute the AND of input bits from all players, where all players receive output. This holds also for other functions with combinatorial properties similar to the AND. Note that for functions that can be securely computed in constant round, this bound is tight up to a constant factor. For $t = 1$ and semi-honest security, we show that $2n$ messages are sufficient to compute any function in non-deterministic log-space, thus the bound is exact for this case. In particular, for $n = 3$ and $t = 1$, 6 messages is necessary and sufficient to compute the AND of three input bits. Note that we use conservative message complexity for our upper bounds and liberal for our lower bound – it may therefore be slightly surprising that we nevertheless get tight bounds.

Next, we consider round complexity: As mentioned, computing any function with unconditional security, in constant round and efficiently in the circuit size of the function is an open problem[1], 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. Also this we prove to be 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 slightly surprising that we need so little involvement from the t selected players. In particular, they might all be corrupt and hence send completely garbled 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 $[n] = \{1, \ldots, n\}$.

[1] Using randomising 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 much towards handling any efficient function efficiently.

We prove security in the model from [Can00] with unconditional security and an adaptive adversary. We consider a synchronous model with point-to-point perfectly secure channels between each pair of parties. 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. 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 partes 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.

It is possible 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 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 gives d_i to P_i. After this the n parties engage in a normal 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 active 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

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

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 $(x_1, \ldots, x_n)$ for f it holds that $f(x_1, \ldots, x_n) = g(M_1(x_1, r), \ldots, M_n(x_n, r))$.

By ϵ-privacy of a PSM we mean that there exists a poly-time simulator S such that for all inputs $(x_1, \ldots, x_n)$ for f, $y = f(x_1, \ldots, x_n)$ and a random sample $r \leftarrow R$ it holds that $(M_1(x_1, r), \ldots, M_n(x_n, r))$ 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 $(M_1(x_1, r), \ldots, M_n(x_n, r))$ learns nothing extra to $y = g(M_1(x_1, r), \ldots, M_n(x_n, r))$.

The PSM model was introduced in [IK97], where they also gave perfectly secure 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 $(M_1(x_1, d_1), \ldots, M_n(x_n, d_n))$ depends only on $f(x_1, \ldots, x_n)$. 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 always possible to compute in poly-time from an output $y = f(x_1, \ldots, x_n)$ an input $(x'_1, \ldots, x'_n)$ such that $y = f(x'_1, \ldots, x'_n)$ then the notions are equivalent for f. The simulator will simply compute $(x'_1, \ldots, x'_n)$, sample $r \leftarrow R$ and output $(M_1(x_1, r), \ldots, M_n(x_n, r))$. We will only use such efficiently invertible functions f in the following.

We also use additive secret sharing of bits strings $x \in \{0,1\}^m$. An additive secret sharings of x between $\mathsf{P}_1, \ldots, \mathsf{P}_n$ consists of sampling shares $s_1, \ldots, s_n \in (\{0,1\}^m)^n$ 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 42 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 any information when there is no message to be sent.

Of course seeing the communicate 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 iP_1 always sends a message to P_2 in round 1, as the absence of the message is a signal.

On the other hand, proving our lower bounds using the measure hinted above is not satisfying. We should be intrigued whether or not using tricks as above will allow more efficient protocols, and we should in lower bounds not count a message which might sometimes be sent for some given input to the protocol towards the complexity of the protocol when run on other inputs. It is not clear this will give sensical bounds, and we should at least worry whether we make it too easy to prove the lower bounds this way.

For similar reasons, when we prove our lower bounds we should not count high communication complexity which occurs with a vanishing probability. 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. So 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 therefore define two measures of message complexity, a conservative one and a liberal one. We use the conservative one when we prove upper bounds. We use the liberal one when we prove lower bounds. It might be surprising in the light of this that we actually manage to prove matching lower and upper bounds.

Definition 1 (Conservative Message Complexity). *Let π be an n-party protocol for a synchronous network. By $\mathsf{Msg}_{\mathtt{con}}(\pi)$ 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 for π such that when π is run with that input, P_i will send a message to P_j in round r with non-zero probability. We let $c_{r,i,j} = 0$ otherwise. We let*

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

Definition 2 (Liberal Message Complexity). *Let π be an n-party protocol for a synchronous network. By $\mathsf{Msg}_{\mathtt{lib}}(\pi)$ we denote the liberal message complexity of π. 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*

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

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

We extend the above notion 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 by ∞.

4 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.

4.1 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 1. *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 receive 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 parties 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 $\pi_{\texttt{hon-maj}}$ the state of P_i in $\pi_{\texttt{hon-maj}}$ is secret shared among the parties in J. Each round in $\pi_{\texttt{hon-maj}}$ 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 $\pi_{\texttt{hon-maj}}$ has 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 $\pi_{\texttt{pre-pro}}$ and $\pi_{\texttt{hon-maj}}$ 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 $\pi_{\texttt{hon-maj}}$ 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 $\pi_{\texttt{hon-maj}}$. Since there are at most t corrupted parties we can then appeal to the security of $\pi_{\texttt{hon-maj}}$.

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. □

4.2 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 1. *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 send a message to each of the parties $\mathsf{P}_1, \ldots, \mathsf{P}_{n-t}$ in the first round and the send or receive 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 when then 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. □

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

Corollary 1. *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 send one message to each of the parties* $\mathsf{P}_1, \ldots, \mathsf{P}_{n-t}$ *in the first round and otherwise have no communication.*

4.3 Individual Round Complexity, Secure Function Evaluation

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

Theorem 2. *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 send one message to the* $n - t$ *other parties in parallel and then later each receive 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 parties n' parties. Denote this protocol by π_{RB}. Set $n' = n - t$. We are going to let the parties I run π_{RB} to compute a particular function g derived from f. In doing that they will implement the broadcast channel using $\pi_{\mathtt{broad}}$ from Corollary 1 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 $\pi_{\mathsf{RB}}(X_1, \ldots, X_{n-t})$.
4. When $\mathsf{P}_i \in I$ learns the output $y = \pi_{\mathsf{RB}}(X_1, \ldots, X_{n-t})$ 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 π_{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 are a majority of honest parties in I. Hence the honest parties in J will also learn the correct output y. □

4.4 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 $t = 1$.

Theorem 3. *For every poly-time n-party function f in non-deterministic log-space, 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 $\mathsf{Msg}(\pi) = 2n$.*

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

Let $(R, M_1, \dots, M_{n-1})$ 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 $i = 2, \dots, n-1$.
3. For $i = 1, \dots, n-1$, party P_i sends $m_i = M_i(x_i, r)$ to P_n.
4. P_n outputs $y = g(m_1, \dots, m_{n-1})$.

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

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

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

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

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 $y = f(x_1, \dots, x_n)$. We first present and analyse a simple solution and then later modify it slightly. 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

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

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 $(R, M_1, \ldots, M_{n-1})$ by 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 another message reduction trick. We are going to reduce the $2(n-2)$ messages in Steps 4 and 5 to just $n-1$ messages. We replace the two steps 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. Then P_1 sends $(p_2, \ldots, p_{n-1})$ to P_n. This can be done in Step 2 above and therefore does not add another message.
2. P_1 sends $(r, p_2, \ldots, p_{n-1})$ to P_2.
3. Then for $i = 2, \ldots, n-1$ party P_i receives $(r, c_2, \ldots, c_{i-1}, p_i, p_{i+1}, \ldots, p_{n-1})$ from P_{i-1} and sends $(r, c_2, \ldots, c_{i-1}, c_i, p_{i+1}, \ldots, p_{n-1})$ to P_{i+1}, where $c_i = M_i(x_i, r) \oplus p_i$, except that P_{n-1} does not send r along 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 $m_i = c_i \oplus p_i$.

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 leaks no information.

It is easy to count that the total number of messages sent is $2n$. □

From Theorem 4 we have the lower bound $n(t+3)/2$ on the message complexity when t is odd. If we set $t = 1$ in this bound we get $n(1+3)/2 = 2n$, matching the upper bound of Theorem 3. We leave it as an open problem to find matching bounds for any $t > 1$.

5 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.

5.1 Message Complexity

We first prove a lower bound on the message complexity of secure function evaluation secure against t semi honest corruptions. We prove the bound for the AND function, but it easily generalises to most non-trivial functions.

Theorem 4. *Let f be the n-party function where the input of P_i is a bit b_i and $f(b_1, \ldots, b_n) = b_1 \wedge \cdots \wedge b_n$. Let π be a protocol which securely computes f with statistical correctness and statistical privacy against t semi-honest corruptions, where $t > 0$ and $n \geq 2t + 1$. Then*

$$\mathsf{Msg}_{\mathtt{lib}}(\pi) \geq \lceil (n(t+3) - 1)/2 \rceil \ .$$

Another way of phrasing the bound is that when n is even or t is odd, then $\mathsf{Msg}_{\mathtt{lib}}(\pi) \geq n(t+3))/2$ and otherwise $\mathsf{Msg}_{\mathtt{lib}}(\pi) \geq n(t+3))/2 - 1$.

Proof. For each party P_i in the protocol, define a counter c_i initialised to 0. Whenever P_i sends or receives a message, increment the counter c_i. Since we are not interested in round complexity we can assume that in each round at most one party sends at most one message[2]. We can simply say that P_i may send a message to P_j only in rounds r where $r \bmod n^2 = (i-1) + (j-1)n$.

We use *point in a protocol* to designate a round r at the time between the message that might have been sent in round $r-1$ was received and before the message sent in round r is sent. This means that at a point in a protocol, no message is in transit. Note that this means that at a point in a protocol $(\sum_{i=1}^{n} c_i) \bmod 2 = 0$. We use the round numbers r to denote points in protocols.

A *communication pattern* C is an ordered list of pairs of players specifying sent a message to who and in which ordered this happened. As an example, if no messages are sent in rounds 1 and 2 and P_1 sends a message to P_7 in in round 3 and no message is sent in round 4 and P_5 sends a message to P_2 in in round 5, then the communication pattern is $((1,7),(5,2))$. For a given round r a a given run of a protocol we use C_r to denote the communication pattern of the run protocol up to round r. Notice that the counters c_i for all parties at round r can be computed from just C_r. Is it just the number of times i occur in a pair in the list C_r. We denote this number by $c_i(C_r)$.

We say that a point r in a protocol with communication pattern C is *premature* if there exists P_i such that $c_i(C_r) < t+1$. We use r_0 to denote the last premature point, letting $r_0 = \infty$ if all points are premature [3]. We say that P_j is *abandoned* in C if he does not receive a message after point r_0. Notice that it can be computed from just the communication pattern C whether P_j was

[2] Our result does cover protocols that send several messages in one round, the assumption made here is simply a way to enforce an order in which we will consider the messages in our proof.

[3] The intuition behind the terminology is that, at a premature point, we should not be able to compute the output yet: if a player has only talked to t players they could all be corrupt and hence his input should not be uniquely defined.

abandoned. We use $\mathrm{Ab}(C) \subseteq \{1, \dots, n\}$ to denote the set of parties that where abandoned in C.

We now argue that if a protocol never abandons anyone, then the complexity is as stated in the theorem. This follows since, in every communication pattern that occurs, all parties receive at least one message after point r_0. Since no message is in transit at point r_0, it follows that at least n messages were sent after point r_0. Before point r_0 we had that there is at most one i for which $c_i = t$ and for $j \neq i$ it holds that $c_j \geq t+1$. The sum of the counters is therefore at least $(n-1)(t+1)+t = n(t+1)-1$. If either n or $t+1$ is even, then this number is odd, which is impossible at any point in a protocol, so since it is a lower bound, we can lift it to $n(t+1)$ when n is even or t is odd. The number of message before r_0 is therefore $\lceil (n(t+1)-1)/2 \rceil$. This gives the desired bound.

We now show that a protocol with perfect correctness (but statistical security) does not abandon anyone, except with a vanishing probability. At the end of the proof, we generalise to statistical correctness. In the analysis we fix n, which we can do as it is the case that if there exists a protocol contradicting the bound, we can fix one such protocol, making n a constant even if we let the statistical security parameter s grow. Assume that the statistical distance between simulation and real protocol is 2^{-s} and let $s = 5\sigma$ for a secondary parameter σ.

In computing the expected communication complexity we can sum the expected complexity contributed by each communication pattern C, as

$$\mathrm{E}_{\boldsymbol{r}}[\mathsf{Msg}(\pi, \boldsymbol{x}, \boldsymbol{r})] = \sum_C \Pr[C]\, \mathrm{E}_{\boldsymbol{r}}[\mathsf{Msg}(\pi, \boldsymbol{x}, \boldsymbol{r}) | C] = \sum_C \Pr[C] |C| \ ,$$

where $\Pr[C]$ is the probability of observing C in a random run of $\pi(\boldsymbol{x})$.

We want to prove that $\sum_C \Pr[C]|C| \geq \lceil (n(t+3)-1)/2 \rceil - \mathrm{negl}(s)$. We have that

$$\sum_C \Pr[C]|C| = \sum_{C \in \mathcal{C}} \Pr[C]|C| + \sum_{C \notin \mathcal{C}} \Pr[C]|C| \leq \sum_{C \in \mathcal{C}} \Pr[C]|C| + \lceil (n(t+3)-1)/2 \rceil (\sum_{C \notin \mathcal{C}} \Pr[C]) \ .$$

It is there sufficient to prove that $\sum_{C \notin \mathcal{C}} \Pr[C] \geq 1 - \mathrm{negl}(s)$. To prove this it is by the law of total probability sufficient to prove that $\sum_{C \in \mathcal{C}} \Pr[C] \leq \mathrm{negl}(s)$. Since we can assume without loss of generality that n is a constant (in s) we can therefore also assume all communication patterns in $\mathcal{C}$ have a length which is bounded by a constant in s, namely $\lceil (n(t+3)-1)/2 \rceil$. Hence $|\mathcal{C}|$ is a constant in s. Therefore, to prove that $\sum_{C \in \mathcal{C}} \Pr[C] \leq \mathrm{negl}(s)$, it is sufficient to prove that for all $C \in \mathcal{C}$ it holds that $\Pr[C] \leq \mathrm{negl}(s)$, as a sum of constantly many negligible functions is again negligible. So, fix any $C \in \mathcal{C}$.

Recall that we proved above that if no party is abandoned in C, then $|C| \geq \lceil (n(t+3)-1)/2 \rceil$. Hence, if $C \in \mathcal{C}$, then *some* party P_j is abandoned in C. For an input $\boldsymbol{x}$ and a communication pattern $C \in \mathcal{C}$ for a complete run of the protocol and a party P_j, let $E_{\boldsymbol{x},C,\mathsf{P}_j}$ be the event that in a random run of $\pi(\boldsymbol{x})$ that C occurs and P_j is abandoned in C. Let $p_{\boldsymbol{x},C,\mathsf{P}_j}$ be the probability of this event. We have that $\Pr[C] \leq \sum_{\mathsf{P}_j} p_{\boldsymbol{x},C,j}$. Since we again sum over only a constant number of terms it is therefore now sufficient to prove that each $p_{\boldsymbol{x},C,\mathsf{P}_j} \leq \mathrm{negl}(s)$. So, for the sake of contradiction, assume that there exists $(\boldsymbol{x}, C, \mathsf{P}_j)$ such that

$$p_{\boldsymbol{x},C,\mathsf{P}_j} \geq 2^{-\sigma} \ .$$

We first prove that this means that $p_{\boldsymbol{x},C,\mathsf{P}_j}$ must be almost this large for *any* inout $\boldsymbol{x}$. This follows because it must hold for all pairs of input $\boldsymbol{x}^0$ and $\boldsymbol{x}^1$ that

$$|p_{\boldsymbol{x}^0,C,\mathsf{P}_j} - p_{\boldsymbol{x}^1,C,\mathsf{P}_j}| \leq 2^{-4\sigma} \ .$$

If this was not the case, we could guess which input is used without corrupting anyone: run π on either $\boldsymbol{x}^0$ or $\boldsymbol{x}^1$ and observe if C occurs and P_j is abandoned. If this event happens, guess that the input was the one that leads to the largest probability for abandoning, and otherwise guess uniformly at random. This would allow to distinguish with advantage $2^{-4\sigma} > 2^{-s}$ over a random guess and would contradict privacy against 0 semi-honest corruptions.

Let P_i be the party who has communicated with at most t parties at point r_0 in C. Use T to denote the set of those parties. Use K to denote the set of parties that are not P_i nor in T. Below we will always give the same inputs to all parties in T and we always give the same inputs to all parties in K. For three bits a, b and c we represent a full input by abc, where a is given to P_i, b is given to all parties in T and c is given to all parties in K.

We say that P_i is *undefined* if it is the case that at point r_0, he can locally compute views that are consistent with both $b_i = 0$ and $b_i = 1$ and his communication with T. Notice the both P_i and the parties in T can compute from the communication between P_i and T whether P_i is undefined. In particular, if P_i is not undefined, the parties T can compute the input of P_i with certainty. Use U to denote the event that P_i is undefined.

We will say that K is *undefined* if it is the case that at point r_0, one can compute joint views for them that are consistent with both $c = 0$ and $c = 1$ and their communication with T. Notice that if there is no communication between P_i and K, then the parties in T can compute from their joint communication with the parties in K whether K is undefined, so if K did not communicate with P_i and K is not undefined then T can compute whether the inputs of the parties in K were 0 or 1. Use V denote the event that K is undefined.

We break the analysis into two, one where $\mathsf{P}_j \in T \cup \{\mathsf{P}_i\}$ and one where $\mathsf{P}_j \in K$.

Assume first that $\mathsf{P}_j \in T \cup \{\mathsf{P}_i\}$. As we saw above,

$$\Pr[E_{010,C,\mathsf{P}_j}] \geq 2^{-\sigma} - 2^{-4\sigma} \geq 2^{-2\sigma} \ .$$

From this it follows that

$$\Pr[E_{010,C,\mathsf{P}_j} \wedge V] \geq 2^{-2\sigma} - 2^{-4\sigma} \geq 2^{-3\sigma} \ .$$

To see this, note that if E_{010,C,P_j} happens, then P_i did not communicate with K so if V does not occur, then the parties in T can compute the input of the parties in K, which they are not allowed to be able to with advantage $2^{-4\sigma} > 2^{-s}$ over random guessing when the input of P_i is 0. Namely, switching $c = 0$ to $c = 1$ does not change the output in the ideal model when $a = 0$, so if the view changed significantly in the real protocol it would give an attack. From this it follows that

$$\Pr[E_{110,C,\mathsf{P}_j} \wedge V] \geq 2^{-3\sigma} - 2^{-4\sigma} \geq 2^{-4\sigma} \ .$$

If this was not the case we could run with input either 110 or 010 and corrupt the parties in T. If C occurs and P_j is abandoned, then make a guess at $a = 0$ if V occurs. Otherwise guess $a = 1$. If C does not occur or P_j is not abandoned, then make a random guess. This gives advantage at least $2^{-4\sigma} > 2^{-s}$. Note that that when E_{110,C,P_j} occurs and V occurs, we can at point r_0 recompute a local view of the parties in K consistent with input $c = 1$ which would give a global run of the protocol consistent with input 111. However, when the input is 110 party P_j should output 0 and when the input is 111, then P_j should output 1. However, by assumption of P_j being abandoned and $\mathsf{P}_j \notin K$, party P_j does not receive any input after point r_0 and neither does it have its input changed when we go from 110 to 111 and therefore cannot switch its output from 0 to 1.

Assume then that $\mathsf{P}_j \in K$. We start again from $\Pr[E_{010,C,\mathsf{P}_j}] \geq 2^{-2\sigma}$ and conclude that $\Pr[E_{010,C,\mathsf{P}_j} \wedge U] \geq 2^{-3\sigma}$, because when U does not occur, then the parties in T can compute the input of P_i which violates privacy since inside this event, the output is always 0. From this it follows that $\Pr[E_{011,C,\mathsf{P}_j} \wedge U] \geq 2^{-4\sigma}$. If this was not the case we could run with input either 011 or 010 and corrupt the parties in T and violate the privacy of the parties in K. Note that when E_{011,C,P_j} occurs and U occurs, we can at point r_0 recompute a local view of P_i consistent with input $a = 1$ which would give a global run of the protocol consistent with input 111. However, when the input is 011 party P_j should output 0 and when the input is 111, then P_j should output 1. However, by assumption of P_j being abandoned and $\mathsf{P}_j \in K$, party P_j does not receive any input after point r_0 and neither does it have its input changed when we go from 011 to 111 and therefore cannot switch its output from 0 to 1.

We finally argue how to generalise to statistical correctness. The only change needed is to change the definition of *undefined* to mean that not only should there be views explaining both input 0 and 1. The fraction of random tapes explaining input 0 should be close to the fraction of random tapes explaining input 1, say at most $2^{-5\sigma}$ apart (and now starting from $s = 6\sigma$). If they are furthermore apart, one can still get a better guess at the input of the parties with advantage $2^{-5\sigma}$. With this definition, when we look at the case where we for instance switch from input 011 to 111 and know that P_j is undefined and cannot have different output distributions in the two cases, we can additionally from

$$\Pr[E_{011,C,\mathsf{P}_j} \wedge U] \geq 2^{-4\sigma}$$

conclude that

$$\Pr[E_{111,C,\mathsf{P}_j} \wedge U] \geq 2^{-4\sigma} - 2^{-5\sigma} \geq 2^{-5\sigma}$$

as there are almost as many random tapes of P_i consistent with the communication with T and inputs 0 or 1. This means that the output of P_i for one of the two inputs must be at least $2^{-5\sigma}/2 < 2^{-s}$ away from the correct output. $\square$

5.2 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 message, each receive a message from a subset of the parties not in M_π.

Theorem 5. *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 4, which says that 6 messages is required.

Theorem 6. *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 message 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 ang 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 alway compute the correct result with good probability when its view is independent of a. $\square$

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.

[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.

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

[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.

[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.