Lower Bounds for Secure Multiparty Computation Without Broadcast

Ran Cohen* Iftach Haitner† Eran Omri‡

March 2, 2015

Abstract

A major challenge in the study of cryptography is characterizing the necessary and sufficient assumptions required to conduct a given cryptographic task. The focus of this work is the necessity of a *broadcast channel* for secure multiparty computation, involving three or more parties. We show that if one third of the parties might be corrupted, broadcast is necessary for securely computing any symmetric n-party functionality that is not an $\lceil \frac{n}{3} \rceil$-*junta*; a functionality is a k-junta, if *any* k-size subset of its input variables can be set to determine its output. The above immediately yields that securely computing (in the presence of one-third corruptions) non-junta symmetric functionalities, like Boolean XOR, requires broadcast. It also shows the necessity of broadcast for securely computing symmetric *random* functionalities like coin-flipping protocols (in which all honest parties must output a common value). The above characterization is tight for three-party symmetric functionalities that can be securely computed *with* broadcast (e.g., Boolean OR), since Cohen and Lindell [Asiacrypt '14] showed that such three-party 1-junta symmetric functionalities, can be securely computed *without* broadcast.

Our proof employs a novel extension of the well-known *hexagon* argument of Fischer et al. [PODC '85], for the case of non-junta randomized functionalities.

Keywords: broadcast; point-to-point communication; multiparty computation; juntas; fairness; coin flipping; impossibility result.

*Department of Computer Science, Bar-Ilan University. E-mail: cohenrb@cs.biu.ac.il.

†School of Computer Science, Tel Aviv University. E-mail: iftachh@cs.tau.ac.il.

‡Department of Computer Science and Mathematics, Ariel University. E-mail: omrier@ariel.ac.il.

Contents

1 Introduction

Broadcast (introduced by Lamport et al. [15] as the Byzantine Generals problem), allows any party to deliver a message of its choice to all honest parties, such that all honest parties will receive the same message, even if the broadcasting party is corrupted. Broadcast is an important resource for implementing secure multiparty computation (MPC) of more than two parties. Indeed, much can be achieved when broadcast is available (hereafter, the broadcast model); in the computational setting, assuming an arbitrary number of corruptions, for every efficient functionality we have MPC guaranteeing security with abort[1] [10, 19], whereas for some functionalities we can achieve security without abort, (e.g., Boolean OR and three-party majority [12]), or p-security[2] without abort (e.g., three-party coin-flipping protocols [13]). In the information theoretic setting, for every efficient functionality we have MPC guaranteeing security without abort against any minority of corrupted parties [18].

The above drastically changes in the setting where broadcast in not available, and specifically when assuming that the parties are (only) connected via a peer-to-peer network (hereafter, the point-to-point model); if less than one third of the parties are corrupted, any efficient functionality can be securely computed without abort [2, 4]. If one third of the parties (or more) might be corrupted, then some important functionalities *cannot* be securely computed (e.g., Byzantine agreement [17, 15] and three-party majority [6]), whereas other functionalities (e.g., *weak* Byzantine agreement [8] and Boolean OR [6]) can be securely computed assuming an arbitrary number of corruptions. The characterization of many other functionalities, however, was unknown. For instance, it was unknown whether coin-flipping or the Boolean XOR functionality can be securely computed, even when only assuming honest majority.

In the following text, we focus on the synchronous model of communication: the protocol proceeds in rounds, and in each round, a party may send and receive a message to/from any other party. We only consider protocols involving three parties or more, as broadcast is equivalent to the point-to-point model in the two-party case.

1.1 Our Result

A protocol is *t-consist*, if in any execution of the protocol in which at most t parties are corrupted, *all* honest parties output the same value. Our main result is the following attack on consist protocols.

Theorem 1.1 (main theorem, informal)**.** *Let $n \geq 3$ and let π be a polynomial-time n-party, $\lceil \frac{n}{3} \rceil$-consist protocol in the point-to-point model. Then, for some value y^*, there exists an efficient adversary that by controlling any $\lceil \frac{n}{3} \rceil$-size subset of the parties, can force the remaining honest parties to output y^*. The above extends to* expected *polynomial-time protocols, and to protocols that only guarantee consistency to hold with high probability.*

The proof of Theorem 1.1 uses a novel extension of the hexagon argument of Fischer et al. [7], that was used for proving the impossibility of reaching (strong and weak) Byzantine agreement in the point-to-point model (in face of one-third corruptions). Specifically, the approach of [7] is useful for attacking *randomized* consist protocols in which a subset of the honest parties (might)

[1] The ideal "implementation" of this functionality, the security of the real implementation is compared to, allows the malicious parties to abort prematurely after getting their part of the output.

[2] The implementation level of security is p-indistinguishable from the ideal implementation of this functionality that does not allow premature aborts.

determine the common output (e.g., Byzantine agreement), or *deterministic* consist protocols (e.g., deterministic protocols computing weak Byzantine agreement). Our approach is applicable for attacking *any randomized* consist protocol.

An immediate application of Theorem 1.1 regards coin-flipping protocols. A coin-flipping protocol [3] allows the honest parties to jointly flip an unbiased coin, where even a coalition of cheating (efficient) parties cannot bias the outcome of the protocol by too much. Our focus is on coin flipping with fairness, where the honest parties *must* output a bit. As an immediate application of Theorem 1.1 we get the following result.

Corollary 1.2 (exists no many-party coin-flipping in the point-to-point model, informal)**.** *In the point-to-point model, there exists no $(n \geq 3)$-party coin-flipping protocol that guarantees non-trivial bias (i.e., smaller than $\frac{1}{2}$) against an efficient adversary controlling one third of the parties.*

It is important to note that in the broadcast model, coin flipping can be securely computed if and only if honest majority exists [5], and can be p-securely computed (the value of p depends on the number of honest and corrupted parties) even when no honest majority is assumed [5, 1, 13].

A different application of Theorem 1.1 is the following characterization of symmetric functionalities (i.e., all parties get the same output value). A functionality is a *k-junta*, if there exists a value y^* (the value of the junta) such that *any* k-size subset of the functionality's input variables, can be manipulated to make the output of the functionality be y^* (e.g., the Boolean OR functionality is a one-junta of value 1).

Corollary 1.3 (Informal)**.** *Let $n \geq 3$. A symmetric n-party functionality that can be securely computed*[3] *in the point-to-point model against $\lceil \frac{n}{3} \rceil$ corruptions, is an $\lceil \frac{n}{3} \rceil$-junta.*

Cohen and Lindell [6] (following Fitzi et al. [8]) showed that any 1-junta functionality (e.g., Boolean OR) that can be securely computed in the broadcast model, can be securely computed in the point-to-point model. Corollary 1.3 yields that for three-party symmetric functionalities, the condition considered by Cohen and Lindell [6] is also necessary.

Corollary 1.4 (Informal)**.** *Let f be a three-party symmetric functionality. Then f can be securely computed in the point-to-point model, if and only if f is a 1-junta that can be securely computed in the broadcast model.*

1.2 Our Technique

We show that the following holds in the point-to-point model. For any efficient consist protocol there exists an efficient adversary that by controlling a third of the protocol's parties, can make the honest parties to output a predetermined value. We show the proof idea for three-party protocols with a single corrupted party, and the generalization to n-party protocols with $\frac{n}{3}$ corrupted parties is standard.

Let $\pi = (\mathsf{A}, \mathsf{B}, \mathsf{C})$ be an efficient consist three-party protocol, and let q be its round complexity on inputs of fixed length κ. Consider the following ring network $\mathsf{R} = (\mathsf{A}^1, \mathsf{B}^1, \mathsf{C}^1, \ldots, \mathsf{A}^q, \mathsf{B}^q, \mathsf{C}^q)$, where each consecutive parties, as well as the first and last, are connected via a private channel, and party P^j, for $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$, has the code of P (see Figure 1).

Consider an execution of R on input $\boldsymbol{w} = (w^1_{\mathsf{A}}, w^1_{\mathsf{B}}, w^1_{\mathsf{C}}, \ldots, w^q_{\mathsf{A}}, w^q_{\mathsf{B}}, w^q_{\mathsf{C}}) \in (\{0,1\}^\kappa)^{3q}$ (i.e., party P^i has input w^i_{P}, containing its actual input and random coins). A key observation is that the

[3] According to the real/ideal paradigm, in which the ideal adversary cannot abort the computation.

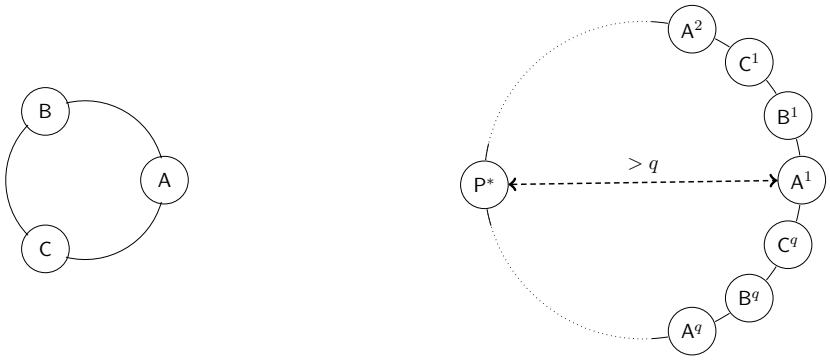

Figure 1: The original 3-party protocol $\pi = (\mathsf{A}, \mathsf{B}, \mathsf{C})$ is on the left. On the right is the q-Ring — q copies of π concatenated. Communication time between parties of opposite sides is larger than $3q/2 > q$.

point of view of the party A^j, for instance, in such an execution, is a *valid* view of the party A on input w^j_{A} in an interaction of π in which B acts honestly on input w^j_{B}. And it is also a valid view of A, on input w^j_{A}, in an interaction of π in which C acts honestly on input $w^{j-1 \pmod q}_{\mathsf{C}}$. Hence, the consistency of π yields that any consecutive pair of parties in R output the same value, and thus *all* parties of R output the same value.

Assume for concreteness that we would like to attack the parties $\{\mathsf{A}, \mathsf{B}\}$. The efficient adversary D first selects a value $\boldsymbol{w} \in (\{0,1\}^\kappa)^{3q}$, emulates an execution of R on $\boldsymbol{w}$, and sets y^* to be the output of the party $\mathsf{P}^* = \mathsf{A}^{q/2}$ in this execution. To interact with the parties $\{\mathsf{A}, \mathsf{B}\}$ in π, D emulates an execution of R, in which all but $\{\mathsf{A}^1, \mathsf{B}^1\}$ have their inputs according to $\boldsymbol{w}$, and $\{\mathsf{A}, \mathsf{B}\}$ take (without knowing that) the roles of $\{\mathsf{A}^1, \mathsf{B}^1\}$.

We claim that the output of $\{\mathsf{A}, \mathsf{B}\}$ under the above attack is y^*. Observe that the emulation of R, induced by the interaction of D with $\{\mathsf{A}, \mathsf{B}\}$, is just a valid execution of R on some input $\boldsymbol{w}'$ (unknown to the adversary). Hence, all parties in R (including $\{\mathsf{A}, \mathsf{B}\}$) output the same value at the end of this emulation. Since the execution of R ends after at most q rounds, and the number of communication links between $\{\mathsf{A}^1, \mathsf{B}^1\}$ and P^* is $\approx 3q/2 > q$, the actions of $\{\mathsf{A}^1, \mathsf{B}^1\}$ have *no effect* on the view of P^*. In particular, the output of P^* in the attack is also y^*, and by the above this is also the output of $\{\mathsf{A}, \mathsf{B}\}$.

1.2.1 Expected Polynomial-Time Protocols

The above attack works perfectly if π runs in (strict) polynomial time. For expected polynomial-time protocols, one has to work slightly harder to get (almost) as good attack.

Let q be the expected round complexity of π. That is, an honest party of π halts after q rounds in expectation, regardless of what the other parties do, where the expectation is over its random coins. Consider the ring $\mathsf{R} = (\mathsf{A}^1, \mathsf{B}^1, \mathsf{C}^1, \ldots, \mathsf{A}^m, \mathsf{B}^m, \mathsf{C}^m)$, for $m = 2q$. By Markov bound, in a random execution of R, a party halts after m rounds with probability at least $\frac{1}{2}$.

The attacker D against $\{\mathsf{A}, \mathsf{B}\}$ is defined as follows. For choosing a value for y^*, it emulates an execution of R on uniform inputs and random coins. If the party $\mathsf{P}^* = \mathsf{A}^{m/2}$ halts in at most m rounds, D sets y^* to be P^*'s output, and continues to the second stage of the attack. Otherwise, it emulates R on a new random input. Note that in k attempts, algorithm D finds a good execution with probably (at least) $1 - 2^{-k}$. After finding y^*, algorithm D continues as in the strict polynomial case discussed above.

The key observation here is that in the emulated execution of R, induced by the interaction of D with $\{\mathsf{A}, \mathsf{B}\}$, the party P^* *never* interacts in more than m communication rounds. Therefore, again, being far from $\{\mathsf{A}^1, \mathsf{B}^1\}$, their actions do not effect P^* in the first m rounds, and thus do not effect it at all. Hence, P^* outputs y^* also in the induced execution, and therefore also the parties $\{\mathsf{A}, \mathsf{B}\}$ do.

1.3 Additional Related Work

In their seminal work, Lamport et al. [15] defined the problem of simulating a broadcast channel in the point-to-point model in terms of the Byzantine agreement (BA) problem.[4] They showed that a solution to the problem exists if and only if more than two thirds of the parties are honest. If the model is augmented with a public key infrastructure (PKI), then a solution exists for any number of corrupted parties (so called authenticated BA). Lamport [14] considered the weak Byzantine agreement (wBA) problem, in which all honest parties must agree on a value, and if all parties are honest and have the same input, they should agree on this common input, and showed that deterministic protocols of finite running time, cannot compute this weak variant of agreement in face of one-third corruptions.

Fischer et al. [7] presented a much simpler impossibility proof for the above mentioned problems (and other distributed consensus tasks). Their idea is to assume a protocol exists for the three-party case, and then compose multiple copies of this protocol into a ring system that contains an internal conflict. Since such ring system cannot exist, it follows that the three-party protocol does not exist. For BA, two copies of the protocol are needed, and the composed ring system forms a hexagon. For wBA, the ring must be large enough so that information from one node cannot traverse the network. Their argument for the latter case crucially relies on the protocol being deterministic.

We note that the impossibility proofs from [7] extend also to public coins protocols, where parties have access to a common random string. It follows that coin flipping is not sufficient for solving BA (or wBA), and thus the impossibility result for coin flipping stated in Corollary 1.2 is not implied by the aforementioned impossibility of Byzantine agreement.

Fitzi et al. [8] presented a probabilistic protocol that securely computes wBA against any number of corrupted parties. Cohen and Lindell [6] analyzed the relation between security in the broadcast model and security in the point-to-point model, and showed (using the protocol from [8]) that any 1-junta functionality that can be securely computed in the broadcast model, can be securely computed in the point-to-point model. They also showed that some (non-junta) functionalities, e.g., three-party majority, that can be computed in the broadcast model cannot be securely computed in the point-to-point model, since they imply the existence of broadcast.

Lindell et al. [16] showed that when $t \geq \frac{n}{3}$, authenticated BA cannot be securely realized under concurrent or parallel (stateless) composition. Goldwasser and Lindell [11] presented a weaker definition for MPC without agreement, in which non-unanimous abort is permitted, i.e., some of the honest parties may receive output while other honest parties might abort, and showed MPC in the point-to-point model, assuming an arbitrary number of corrupted parties, that compose in parallel or concurrently.

[4] The Byzantine agreement problem and the Byzantine Generals problem are essentially equivalent.

Paper Organization

Basic definitions can be found in Section 2. Our attack is described in Section 3, and its applications are given in Section 4. Open problems are discussed in Section 5.

2 Preliminaries

2.1 Notations

We use calligraphic letters to denote sets, uppercase for random variables, lowercase for values, boldface for vectors, and sans-serif (e.g., A) for algorithms (i.e., Turing Machines). For $n \in \mathbb{N}$, let $[n] = \{1, \cdots, n\}$. Let poly denote the set all polynomials and let PPTM denote a probabilistic algorithm that runs in *strictly* polynomial time. A function $\nu \colon \mathbb{N} \mapsto [0,1]$ is *negligible*, denoted $\nu(\kappa) = \mathrm{neg}(\kappa)$, if $\nu(\kappa) < 1/p(\kappa)$ for every $p \in \mathrm{poly}$ and large enough κ.

2.2 Protocols

An n-party protocol $\pi = (\mathsf{P}_1, \ldots, \mathsf{P}_n)$ is an n-tuple of probabilistic interactive TMs. The term *party* P_i refers to the i'th interactive TM. Each party P_i starts with input $x_i \in \{0,1\}^*$ and random coins $r_i \in \{0,1\}^*$. Without loss of generality, the input length of each party is assumed to be the security parameter κ. An *adversary* D is another interactive TM describing the behavior of the corrupted parties. It starts the execution with input that contains the identities of the corrupted parties and their private inputs, and possibly an additional auxiliary input.

The parties execute the protocol in a synchronous network with rushing. That is, the execution proceeds in rounds: each round consists of a *send phase* (where parties send their message from this round) followed by a *receive phase* (where they receive messages from other parties). The adversary is assumed to be *rushing*: it can see the messages the honest parties send in a given round, before determining the messages that the corrupted parties, under its control, send in that round.

In the *point-to-point (communication) model*, which is the one we assume by default, all parties are connected via a *fully connected point-to-point network*. In the *broadcast model*, all parties are given access to a physical broadcast channel in addition to the point-to-point network. The communication lines between parties are assumed to be ideally authenticated but not private (and thus the adversary cannot modify messages sent between two honest parties but can read them).[5] Furthermore, the delivery of messages between honest parties is guaranteed. Finally, we note that we do not assume any trusted preprocessing phase (that can be used to setup a public-key infrastructure, for example).

Throughout the execution of the protocol, all the honest parties follow the instructions of the prescribed protocol, whereas the corrupted parties receive their instructions from the adversary. The adversary is considered to be *malicious*, meaning that it can instruct the corrupted parties to deviate from the protocol in any arbitrary way. At the conclusion of the execution, the honest parties output their prescribed output from the protocol, the corrupted parties output nothing and the adversary outputs an (arbitrary) function of its view of the computation (containing the corrupted parties' view).

[5] If private channels are needed, then privacy can be achieved over authenticated channels by simply using public-key encryption. This works for static corruptions and computationally-bounded adversaries, as we consider in this work.

2.2.1 Time and Round Complexity

We consider both *strict* and *expected* bounds on time and round complexity.

Definition 2.1 (time complexity)**.** *Protocol $\pi = (\mathsf{P}_1, \ldots, \mathsf{P}_n)$ is a T-*time protocol*, if for every $i \in [n]$ and every input $x_i \in \{0,1\}^*$, random coins $r_i \in \{0,1\}^*$, and sequence of messages P_i receives during the course of the protocol, the running-time of an honest party P_i is at most $T(|x_i|)$. If $T \in$ poly, then π is of* (strict) polynomial time*.*

Protocol π has an expected running time T*, if for every $i \in [n]$, every input $x_i \in \{0,1\}^*$ and sequence of messages P_i receives during the course of the protocol, the expected running time of an honest party P_i, over its random coins r_i, is at most $T(|x_i|)$. If $T \in$ poly, then π has* expected polynomial running time*.*

Definition 2.2 (round complexity)**.** *Protocol $\pi = (\mathsf{P}_1, \ldots, \mathsf{P}_n)$ is a q-*round protocol*, if for every $i \in [n]$ and every input $x_i \in \{0,1\}^*$, random coins $r_i \in \{0,1\}^*$, and sequence of messages P_i receives during the course of the protocol, the number of rounds of an honest party P_i is at most $q(|x_i|)$. If $q \in$ poly, then π has* (strict) polynomial round complexity*.*

Protocol π has an expected round complexity q*, if for every $i \in [n]$, every input $x_i \in \{0,1\}^*$ and sequence of messages P_i receives during the course of the protocol, the expected number of rounds of an honest party P_i, over its random coins r_i, is at most $q(|x_i|)$. If $q \in$ poly, then π has* expected polynomial round complexity*.*

3 Attacking Consist Protocols

In this section, we present a lower bound for secure protocols in the point-to-point model. Protocols in consideration are only assumed to have a very mild security property, and we postpone discussing the more standard notion of security to Section 4. Specifically, we require all honest parties to be consistent with each other — to output the same value. We emphasize that in a consist protocol, a party may output the special error symbol $\perp$ (i.e., abort), but it can only do so if all honest parties output $\perp$ as well.

Definition 3.1 (consist protocols)**.** *A protocol π is (δ, t)-*consist *against C-class (e.g., polynomial-time, expected polynomial-time) adversaries, if for any C-class adversary D controlling at most t parties, on security parameter κ, all honest parties output the same value with probability at least $\delta(\kappa)$.*

Equipped with the above definition, we start, Section 3.1, with an attack on consist protocols whose number of rounds is strictly bounded, and then, Section 3.2, move to consist protocols with bound on their *expected* number of rounds.

3.1 Protocols of Strict Running-Time Guarantee

Theorem 3.2. *Let $n \geq 3$, and let π be an n-party, T-time, q-round point-to-point protocol. If π is $(1-\delta, \lceil \frac{n}{3} \rceil)$-consist against $(T_{\mathsf{D}} = 2nqT)$-time adversaries, then there exists a T_{D}-time adversary D such that the following holds. On security parameter κ, D first outputs a value y^*. Next, given the control on $\lceil \frac{n}{3} \rceil$ of the parties, D interacts with the remaining honest parties of π on inputs of length κ, and except for probability at most $\frac{3}{2} \cdot q(\kappa) \cdot \delta(\kappa)$, the output of* every *honest party in this execution is y^*.*

For a polynomial-time protocol that is $(1-\text{neg}, \lceil \frac{n}{3} \rceil)$-consist against PPT adversaries, Theorem 3.2 yields a PPT adversary that by controlling $\lceil \frac{n}{3} \rceil$ of the parties can manipulate the honest parties' output (i.e., making them all to be y^*) with all but negligible probability.

Proof. We focus on the three-party case, and the many-party case follows via standard techniques.[6] We also fix the input length parameter κ and omit it from the notation when its value is clear from the context.

Let $\pi = (\mathsf{A}, \mathsf{B}, \mathsf{C})$, let $m = q$ and assume for ease of notation that m is even. Consider the following ring network $\mathsf{R} = (\mathsf{A}^1, \mathsf{B}^1, \mathsf{C}^1, \ldots, \mathsf{A}^m, \mathsf{B}^m, \mathsf{C}^m)$, where each consecutive parties, as well as the first and last, are connected via a private channel, and party P^j, for $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$, has the code of P. Let $v = \kappa + T(\kappa)$, and consider an execution of R on parties' inputs and random coins $\boldsymbol{w} = (w^1_{\mathsf{A}}, w^1_{\mathsf{B}}, w^1_{\mathsf{C}}, \ldots, w^m_{\mathsf{A}}, w^m_{\mathsf{B}}, w^m_{\mathsf{C}}) \in (\{0,1\}^v)^{3m}$ (i.e., party P^i has input w^i_{P}, containing its actual input and random coins).

A key observation is that the point of view of the party A^j, for instance, in such an execution, is a *valid* view of the party A on input w^j_{A} in an execution of π in which B acts honestly on input w^j_{B}. And is also a valid view of A, on input w^j_{A}, in an execution of π in which C acts honestly on input $w^{j-1 \pmod m}_{\mathsf{C}}$. This observation yields the following consistency property of R.

Claim 3.3. *In an execution of* R *on* $\boldsymbol{w} \leftarrow (\{0,1\}^v)^{3m}$*, parties of distance* k *in* R*, measured in (minimal) number of communication links between them, output the same value with probability at least* $1 - k\delta$*.*

Proof. Consider the pair of neighboring parties $\{\mathsf{A}^j, \mathsf{B}^j\}$ in the ring R (the same argument holds for any neighboring parties). Let D be an adversary controlling the party C of π that interacts with $\{\mathsf{A}, \mathsf{B}\}$ by emulating a uniform execution of R (apart from the role of $\{\mathsf{A}^j, \mathsf{B}^j\}$), and let $\{\mathsf{A}, \mathsf{B}\}$ take (without knowing that) the role of $\{\mathsf{A}^j, \mathsf{B}^j\}$ in this execution. The views of $\{\mathsf{A}, \mathsf{B}\}$ in this emulation has the same distribution as the views of $\{\mathsf{A}^j, \mathsf{B}^j\}$ in a random execution of R. Hence, the $(1-\delta)$-consistency of π yields that A^j and B^j output the same value on a uniform choice of $\boldsymbol{w}$, with probability at least $1 - \delta$. The proof follows by a union bound. □

The adversary D first selects uniformly a value for $\boldsymbol{w}$, and sets y^* to be the output of $\mathsf{P}^* = \mathsf{A}^{m/2}$ in the execution of R on $\boldsymbol{w}$. Assume for concreteness that we would like to attack the parties $\{\mathsf{A}, \mathsf{B}\}$. To interact with $\{\mathsf{A}, \mathsf{B}\}$ in π, D emulates an execution of R, in which all but $\{\mathsf{A}^1, \mathsf{B}^1\}$ have their inputs according to $\boldsymbol{w}$, and $\{\mathsf{A}, \mathsf{B}\}$ take the roles of $\{\mathsf{A}^1, \mathsf{B}^1\}$. The key observation is that the view of party P^* in the emulation induced by the above attack, is the *same* as its view in the execution of R on $\boldsymbol{w}$ (regardless of the inputs of $\{\mathsf{A}, \mathsf{B}\}$). This is true since the execution of R ends after at most m communication rounds. Thus the actions of $\{\mathsf{A}^1, \mathsf{B}^1\}$ have no effect on the view of P^*, and therefore the output of P^* is y^* also in the emulated execution of R. Finally, since the parties' inputs in the emulated execution of R are uniformly distributed, and since the distance between P^* and the pair $\{\mathsf{A}^1, \mathsf{B}^1\}$ is (less than) $\frac{3m}{2}$, Claim 3.3 yields that with probability $1 - \frac{3m}{2} \cdot \delta$, the output of $\{\mathsf{A}, \mathsf{B}\}$ under the above attack is y^*. □

[6] I.e., by considering the three-party protocol resulting from partitioning the parties of the original n-party protocol into three $\frac{n}{3}$-size sets.

3.2 Protocols of Expected Running-Time Guarantee

Theorem 3.4. *Let $n \geq 3$, let z be an integer function and let π be an n-party point-to-point protocol, of expected running-time T and expected round complexity q. If π is $(1-\delta, \lceil \frac{n}{3} \rceil)$-consist against adversaries of expected running-time $T_{\mathsf{D}} = 2n(z+1)qT$, then there exists an adversary D of expected running-time T_{D} such that the following holds. On security parameter κ, D first outputs a value y^*. Next, given the control on $\lceil \frac{n}{3} \rceil$ of the parties, D interacts with the remaining honest parties of π on inputs of length κ, and except for probability at most $6 \cdot q(\kappa) \cdot \delta(\kappa) + 2^{-z(\kappa)}$, the output of* every *honest party in this execution is y^*.*

For an expected polynomial-time protocol that is $(1 - \mathrm{neg}, \lceil \frac{n}{3} \rceil)$-consist against expected polynomial-time adversaries, Theorem 3.2 yields (taking $z = \omega(\log \kappa)$) an expected polynomial-time adversary that by controlling $\lceil \frac{n}{3} \rceil$ of the parties, can manipulate the honest parties' output with all but negligible probability.

Proof. We prove for the three-party case. We fix the input length parameter κ and omit it from the notation when clear from the context.

Let $\pi = (\mathsf{A}, \mathsf{B}, \mathsf{C})$ and let $m = 2q$. Similarly to the proof of Theorem 3.2, we consider the (now double size) ring $\mathsf{R} = (\mathsf{A}^1, \mathsf{B}^1, \mathsf{C}^1, \ldots, \mathsf{A}^m, \mathsf{B}^m, \mathsf{C}^m)$. The attacker D follows in similar lines as those used in the proof of Theorem 3.2. The main difference is that in order to select y^*, D iterates the following for z times. In each iteration, D emulates an execution of the ring R on uniform inputs and random coins,[7] for m communication rounds. If during one of these iterations party $\mathsf{P}^* = \mathsf{A}^{m/2}$ halts, D sets y^* to be its output in this iteration. If after z iterations the value y^* was not set, D outputs $\perp$ and aborts. The rest of the attack is identical to that described in the proof of Theorem 3.2.

We start with the following observation.

Claim 3.5. $\Pr[\mathsf{D}$ *aborts* $] \leq 2^{-z}$.

Proof. By Markov bound, the probability that in a single iteration of D the party P^* does not halt within $m = 2q$ rounds, is at most $\frac{1}{2}$. Therefore, the probability that y^* is not set in all z iterations is at most 2^{-z}. □

Since P^* halts in the iteration that produced y^* within m rounds, its view in the emulated execution of R induced by the attack, is the *same* as in this selected iteration (this holds even though some parties might run for more rounds in the emulated execution). In particular, P^* outputs y^* also in the emulated execution. The proof continues as in the proof of Theorem 3.2, where the only additional subtlety is that it is no longer true that the parties' inputs in the emulated execution induced by the attack are uniformly distributed. Indeed, we have selected a value for $\boldsymbol{w}$ that causes P^* to halt within m rounds. Yet, since in a random execution of R, party P^* halts within m rounds with probability at least $\frac{1}{2}$, the method used to sample $\boldsymbol{w}$ at most doubles the probability of inconsistency in the ring. It follows that the attacked parties $\{\mathsf{A}, \mathsf{B}\}$ output y^* with probability at least $1 - 2 \cdot 3q(\kappa) \cdot \delta(\kappa)$ times the probability that D does not abort, and the proof of the theorem follows. □

[7] Note that now we have no a-priori bound on the number of random coins used by the protocol's parties. Yet, the emulation can be done in expected time nmT.

4 Impossibility Results for Secure Computation

In this section, we give few applications of the attack, presented in Section 3, to secure multiparty computations in the point-to-point model, in the setting where a third of the parties (or more) might be corrupted. In Section 4.1, we show that coin-flipping protocols, in which the honest parties must output a bit, are impossible. In Section 4.2, we show that the only symmetric functionalities that can be securely realized according to the real/ideal paradigm, are those which are $\lceil \frac{n}{3} \rceil$-juntas. For concreteness we focus below on (strict) polynomial-time protocols secure against (strict) polynomial-time adversaries, but all the results readily extend to expected polynomial-time regime.

4.1 Coin-Flipping Protocols

A coin-flipping protocol [3] allows the honest parties to jointly flip an unbiased coin, where even a coalition of cheating (efficient) parties cannot bias the outcome of the protocol by very much. Our focus is on coin flipping with fairness, where the honest parties *must* output a bit.

Definition 4.1. *A polynomial-time n-party protocol π is a (γ, t)-*bias *coin-flipping protocol, if the following holds.*

1. *π is $(1, t)$-consist against* PPT *adversaries.*[8]
2. *When interacting on security parameter κ with a* PPT *adversary controlling at most t corrupted parties, the common output of the honest parties is $\gamma(\kappa)$-close to the being a uniform bit.*[9]

Corollary 4.2. *In the point-to-point model, for $n \geq 3$ and $\gamma(\kappa) < \frac{1}{2} - 2^{-\kappa}$, there exists no n-party, $(\gamma, \lceil \frac{n}{3} \rceil)$-bias coin-flipping protocol.*

Proof. Let π be a point-to-point n-party protocol, that is $(1, \lceil \frac{n}{3} \rceil)$-consist against PPT adversaries. Let D be the algorithm guaranteed by Theorem 3.2, let $Y(\kappa)$ denote $\mathsf{D}(\kappa)$'s output in the first step of the attack, and consider some fixed set of $\lceil \frac{n}{3} \rceil$ corrupted parties of π. Without loss of generality, for infinitely many values of κ it holds that $\Pr[Y(\kappa) = 0] \leq \frac{1}{2}$. Consider the attacker that on security parameter κ, repeats the first step of $\mathsf{D}(\kappa)$ until the resulting value of y^* is non-zero or κ failed attempts have reached, where if the latter happens it aborts. Next, it continues the non-zero execution of D to make the honest parties of π output y^*. It is immediate that for infinitely many values of κ, the common output of the honest parties under the above attack is 0 with probability at most $2^{-\kappa}$, and hence the honest parties' common output is $\frac{1}{2} - 2^{-\kappa}$ far from uniform. Thus, π is not a $(\gamma, \lceil \frac{n}{3} \rceil)$-bias coin-flipping protocol. □

4.2 Symmetric Functionalities Secure According to the Real/Ideal Paradigm

Basic definitions are given in Section 4.2.1, and the impossibility results are stated and proved in Section 4.2.2.

[8] Our negative result readily extends to protocols where consistency is only guaranteed to hold with high probability.
[9] In particular, the honest parties are allowed to abort with probability at most γ.

4.2.1 Model Definition

We provide the basic definitions for secure multiparty computation according to the real/ideal paradigm, for further details see [9].

The security definition is based on the *real/ideal paradigm*, i.e., comparing what an adversary can do in the real execution of the protocol to what it can do in an ideal model, where an uncorrupted trusted party assists the parties. Informally, a protocol is secure if whatever an adversary can do in the real protocol (where no trusted party exists) can be done in the ideal computation. We consider *full security*, where the ideal-model adversary is not allowed to abort the ideal execution.

Functionalities.

Definition 4.3 (functionalities)**.** *An n-party functionality is a random process that maps vectors of n inputs to vectors of n outputs, denoted as $f\colon (\{0,1\}^*)^n \mapsto (\{0,1\}^*)^n$, where $f = (f_1, \ldots, f_n)$. That is, for a vector of inputs $\boldsymbol{x} = (x_1, \ldots, x_n)$, the output-vector is a random variable $(f_1(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x}))$ ranging over vectors of strings. The output for the i'th party (with input x_i) is defined to be $f_i(\boldsymbol{x})$. The functionality f is called* symmetric*, if all parties output the same value, i.e., $f_1 = f_2 = \ldots = f_n$.*

The real world.

Definition 4.4 (real-model execution)**.** *Let f be an n-party functionality, let π be a multiparty protocol for computing f and let κ be the security parameter. Denote by $\mathcal{I} \subseteq [n]$ the set of indices of the parties corrupted by D. Then, the* joint execution of π under $(\mathsf{D}, \mathcal{I})$ in the real model*, on input vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, auxiliary input z to D and security parameter κ, denoted $\mathrm{REAL}_{\pi,\mathcal{I},\mathsf{D}(z)}(\boldsymbol{x}, \kappa)$, is defined as the output vector of $\mathsf{P}_1, \ldots, \mathsf{P}_n$ and D resulting from the protocol interaction, where for every $i \in \mathcal{I}$, party P_i computes its messages according to D, and for every $j \notin \mathcal{I}$, party P_j computes its messages according to π.*

The ideal world. The following definition provides the strongest notion of security we consider, it asserts that the protocol can terminate only when all parties receive their prescribed output. A malicious party can always substitute its input or refuse to participate. Therefore, the ideal model takes this inherent adversarial behavior into account by giving the adversary the ability to do this also in the ideal model. An ideal execution proceeds as follows:

Send inputs to trusted party: Each honest party P_i sends its input x_i to the trusted party. Maliciously corrupted parties may send the trusted party arbitrary inputs as instructed by the adversary. Denote by x'_i the value sent by P_i.

Trusted party answers the parties: If x'_i is outside of the domain for P_i or P_i sends no input, the trusted party sets x'_i to be some predetermined default value. Next, the trusted party computes $f(x'_1, \ldots, x'_n) = (y_1, \ldots, y_n)$ and sends y_i to party P_i for every i.

Outputs: Honest parties always output the message received from the trusted party and the corrupted parties output nothing. The adversary outputs an arbitrary function of the initial inputs $\{x_i\}_{i\in\mathcal{I}}$ and the messages received by the corrupted parties from the trusted party $\{y_i\}_{i\in\mathcal{I}}$.

Definition 4.5 (ideal-model computation)**.** *Let $f\colon (\{0,1\}^*)^n \mapsto (\{0,1\}^*)^n$ be an n-party functionality, let $\mathcal{I} \subseteq [n]$ be the set of indices of the corrupted parties, and let κ be the security parameter. The* joint execution of f under (D, I) in the ideal model*, on input vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, auxiliary input z to D and security parameter κ, denoted $\text{IDEAL}_{f,\mathcal{I},\mathsf{D}(z)}(\boldsymbol{x}, \kappa)$, is defined as the output vector of $P_1, \ldots, P_n$ and D resulting from the above described ideal process.*

Security definition. Having defined the ideal and real models, we can now define security of protocols. The underlying idea of the definition is that the adversary can do no more harm in a real protocol execution than in the ideal model (where security trivially holds). This is formulated by saying that adversaries in the ideal model are able to simulate adversaries in an execution of a protocol in the real model.

Definition 4.6. *Let $f\colon (\{0,1\}^*)^n \mapsto (\{0,1\}^*)^n$ be an n-party functionality, and let π be a protocol computing f. A* protocol π t-securely computes f*, if for every non-uniform polynomial-time real-model adversary D, there exists a non-uniform (expected) polynomial-time adversary S for the ideal model, such that for every $\mathcal{I} \subseteq [n]$ of size $|\mathcal{I}| \leq t$, it holds that*

$$\left\{\text{REAL}_{\pi,\mathcal{I},\mathsf{D}(z)}(\boldsymbol{x},\kappa)\right\}_{(\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1},\kappa\in\mathbb{N}} \overset{\text{c}}{\equiv} \left\{\text{IDEAL}_{f,\mathcal{I},\mathsf{S}(z)}(\boldsymbol{x},\kappa)\right\}_{(\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1},\kappa\in\mathbb{N}}.$$

4.2.2 The Lower Bound

Juntas. A special class of symmetric functionalities are those with the property that every subset of a certain size forms a junta, and can fully determine the output to be some "default value". For example, the multiparty Boolean AND and OR functionalities both have the property that every individual party is a dictator, i.e., a junta of size 1 (for the AND functionality any party can always force the output to be 0, and for the OR functionality any party can always force the output to be 1).

Definition 4.7 (juntas)**.** *An n-party symmetric functionality f is a k-*junta*, if there exists a value y^* (the value of the junta) such that for every k-size subset $\mathcal{I} \subseteq [n]$ there exist inputs $\{x_i\}_{i\in\mathcal{I}}$, such that $f(x_1, \ldots, x_n) = y^*$ for* any *complementing subset of inputs $\{x_j\}_{j\notin\mathcal{I}}$.*

Consider a protocol π that $\lceil \frac{n}{3} \rceil$-securely computes a symmetric n-party functionality f. Since f is symmetric, it immediately follows that π is consist (all parties receive the same output in the ideal model, so this holds also in the real model, except for a negligible probability), and therefore the attack described in Section 3 can be applied on π. That is, there exists an efficient adversary D corrupting a third of the parties, that can force the output of the honest parties of π to be some predetermined value $y^* \in \{0,1\}^* \cup \{\perp\}$. Since π $\lceil \frac{n}{3} \rceil$-securely computes f, there must exist an ideal-model adversary S that can perform the same attack on the computation in the ideal model.

When considering security with abort, the ideal-model adversary S can *always* force the output of the honest parties to be $\perp$ (by sending the abort command to the trusted party), yielding that the above attack might be trivial to achieve (if $y^* = \perp$). But when considering full security, as we do here, S cannot abort the ideal-model computation, rather it has to send (valid) inputs to the trusted party. It follows that S manages to make the output of the honest parties to be y^* by only controlling t of the functionality's inputs. Namely, that f is an $\lceil \frac{n}{3} \rceil$-junta.

Corollary 4.8. *Let $n \geq 3$, let $t \geq \frac{n}{3}$ and let f be a symmetric n-party functionality that can be t-securely computed in the point-to-point model, then f is an $\lceil \frac{n}{3} \rceil$-junta.*

Proof. Let π be a protocol that t-securely computes f in the point-to-point model, and let D be the (uniform) PPT adversary guaranteed from Theorem 3.2. It follows that there exists a non-uniform polynomial-time D' and a value y^*, that by controlling any $\lceil \frac{n}{3} \rceil$-size subset $\mathcal{I} \subseteq [n]$ of the parties, it forces the output of the honest parties to be y^*. Since π t-securely computes f, there exists an ideal-model adversary S that upon corrupting the subset of the of parties indexed by $\mathcal{I}$, can force the output of the honest parties in the ideal-model computation to be y^*. Since all S can do is to select the input values of the corrupted parties, there must exist input values $\{x_i\}_{i \in \mathcal{I}}$ that determine the output of the honest parties to be y^*. Since the above holds for any $\lceil \frac{n}{3} \rceil$-size $\mathcal{I} \subseteq [n]$, it follows that f is an $\lceil \frac{n}{3} \rceil$-junta. □

Cohen and Lindell [6, Thm. 3.2] showed that any 1-junta functionality that can be t-securely computed in the broadcast model, can be t-securely computed in the point-to-point model. Combining their result with Corollary 4.8, we obtain a complete characterization of symmetric three-party functionalities.

Corollary 4.9. *Let f be a symmetric three-party functionality. Then*

1. *f can be 2-securely computed in the point-to-point model, if and only if f is a 1-junta that can be 2-securely computed in the broadcast model.*
2. *f can be 1-securely computed in the point-to-point model, if and only if f is a 1-junta.*

Proof. The first part follows immediately by Corollary 4.8 and Cohen and Lindell [6], and the second part follows by the first one, since every functionality can be computed in the broadcast model assuming honest majority. □

5 Open Problems

The main open problem is to provide a complete characterization of functionalities that can be securely computed in the point-to-point model. Two classes of functionalities, for which the characterization is not known, are presented below:

- Non-symmetric functionalities (i.e., parties do not obtain the same output). For example, can three-party coin flipping be securely computed against a single corruption when only two parties obtain output?
- Symmetric functionalities which are $\lceil \frac{n}{3} \rceil$-junta but not 1-junta. For example, can the following 6-party functionality be 2-securely computed:

$$\begin{aligned} f(x_1, \ldots, x_6) = {} & (x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee (x_1 \wedge x_4) \vee (x_1 \wedge x_5) \vee (x_1 \wedge x_6) \vee \\ & (x_2 \wedge x_3) \vee (x_2 \wedge x_4) \vee (x_2 \wedge x_5) \vee (x_2 \wedge x_6) \vee \\ & (x_3 \wedge x_4) \vee (x_3 \wedge x_5) \vee (x_3 \wedge x_6) \vee \\ & (x_4 \wedge x_5) \vee (x_4 \wedge x_6) \vee \\ & (x_5 \wedge x_6). \end{aligned}$$

References

[1] A. Beimel, E. Omri, and I. Orlov. Protocols for multiparty coin toss with dishonest majority. In *Advances in Cryptology – CRYPTO 2010*, pages 538–557, 2010.

[2] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1–10, 1988.

[3] M. Blum. Coin flipping by telephone. In *Advances in Cryptology – CRYPTO '81*, pages 11–15, 1981.

[4] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC)*, pages 11–19, 1988.

[5] R. Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC)*, pages 364–369, 1986.

[6] R. Cohen and Y. Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. In *Advances in Cryptology – ASIACRYPT 2014*, pages 466–485, 2014.

[7] M. J. Fischer, N. A. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. In *PODC*, pages 59–70, 1985.

[8] M. Fitzi, D. Gottesman, M. Hirt, T. Holenstein, and A. Smith. Detectable byzantine agreement secure against faulty majorities. In *PODC*, pages 118–126, 2002.

[9] O. Goldreich. *Foundations of Cryptography – VOLUME 2: Basic Applications*. Cambridge University Press, 2004.

[10] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 218–229, 1987.

[11] S. Goldwasser and Y. Lindell. Secure computation without agreement. In *DISC*, pages 17–32, 2002.

[12] D. Gordon and J. Katz. Complete fairness in multi-party computation without an honest majority. In *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009*, pages 19–35, 2009.

[13] I. Haitner and E. Tsfadia. An almost-optimally fair three-party coin-flipping protocol. In *Proceedings of the 46st Annual ACM Symposium on Theory of Computing (STOC)*, pages 817–836, 2014.

[14] L. Lamport. The weak byzantine generals problem. *Journal of the ACM*, 30(3):668–676, 1983.

[15] L. Lamport, R. E. Shostak, and M. C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[16] Y. Lindell, A. Lysyanskaya, and T. Rabin. On the composition of authenticated byzantine agreement. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 514–523, 2002.

[17] M. C. Pease, R. E. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, 1980.

[18] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 73–85, 1989.

[19] A. C. Yao. Protocols for secure computations. In *Proceedings of the 23th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 160–164, 1982.