Broadcast-Optimal Two-Round MPC

Ran Cohen[*] Juan Garay[†] Vassilis Zikas[‡]

October 10, 2019

Abstract

An intensive effort by the cryptographic community to minimize the round complexity of secure multi-party computation (MPC) has recently led to optimal two-round protocols from minimal assumptions. Most of the proposed solutions, however, make use of a broadcast channel in every round, and it is unclear if the broadcast channel can be replaced by standard point-to-point communication in a round-preserving manner, and if so, at what cost on the resulting security.

In this work, we provide a complete characterization of the trade-off between number of broadcast rounds and achievable security level for two-round MPC tolerating arbitrarily many active corruptions. Specifically, we consider all possible combinations of broadcast and point-to-point rounds against the three standard levels of security for maliciously secure MPC protocols, namely, security with identifiable, unanimous, and selective abort. For each of these notions and each combination of broadcast and point-to-point rounds, we provide either a tight feasibility or an infeasibility result of two-round MPC. Our feasibility results hold assuming two-round OT in the CRS model, whereas our impossibility results hold given any correlated randomness.

[*]Northeastern University. E-mail: rancohen@ccs.neu.edu.
[†]Texas A&M University. E-mail: garay@cse.tamu.edu.
[‡]School of Informatics, University of Edinburgh & IOHK. E-mail: vzikas@inf.ed.ac.uk.

Contents

1 Introduction

Round complexity is an important efficiency measure of secure multi-party computation protocols (MPC) [66, 39], with a large body of research focusing on how it can be minimized. The "holy grail" in this thread has been two-round protocols, as single-round MPC for a large set of functions cannot be achieved [42]. The first solutions to this problem were based on strong cryptographic assumptions (FHE/iO) [5, 33, 58], whereas more recent results showed how to build two-round MPC protocols resilient to any number of active corruptions from standard assumptions, such as two-round oblivious transfer (OT) [32, 10, 11] or OT-correlation setup and one-way functions (OWF) [34] (we discuss the state of the art in Section 1.1).

The advantage of such two-round MPC protocols, however, is often dulled by the fact that the protocols make use of a broadcast channel in the case of malicious adversaries. Indeed, in practice such a broadcast channel is typically not available to the parties, who instead need to use a broadcast protocol over point-to-point communication for this task. Classical impossibility results from distributed computing imply that any such deterministic protocol tolerating (up to) t corruptions requires $t+1$ rounds of communication [27, 26]; these bounds extend to *randomized* broadcast, showing that termination cannot be guaranteed in constant rounds [18, 51]. Even when considering *expected* round complexity, randomized broadcast would require $\Omega(n/(n-t))$ rounds [29] when the adversary can corrupt a majority of parties (i.e., $t \geq n/2$), and *expected two rounds* are unlikely to suffice for reaching agreement, even with weak guarantees, as long as $t > n/4$ [24] (as opposed to expected *three* rounds [57]). Furthermore, while the above lower bounds consider broadcasting just a single message, known techniques for composing randomized broadcast protocols with non-simultaneous termination require a multiplicative blowup of $c > 2$ rounds [54, 8, 52, 23, 21].

The above state of affairs motivated a line of work investigating the effect in the round complexity of removing the assumption of broadcast from two-round MPC protocols [48, 50, 59, 2, 4]. In order to do so, however, one needs to settle for weaker security definitions. In other words, one needs to trade off security guarantees for lower round complexity.

In this work, we fully characterize the optimal trade-off between security and use of broadcast in two-round MPC protocols against a malicious adversary who corrupts any number of parties: In a nutshell, for each of the three standard security definitions that are achievable against such adversaries in the round-unrestricted setting—namely, security with *identifiable*, *unanimous*, or *selective* abort—we provide protocols that use the provably minimal number of broadcast rounds (a broadcast round is a round in which at least one party broadcasts a message using a broadcast channel). Our positive results assume, as in the state-of-the-art solutions, existence of a two-round oblivious transfer (OT) protocol in the CRS model (alternatively, OT-correlation setup and OWF), whereas our impossibility results hold for any correlated randomness setup.

1.1 Background

Starting with the seminal works on MPC [66, 39, 9, 17, 64], a major goal has been to strike a favorable balance between the resources required for the computation (e.g., the protocol's round complexity), the underlying assumptions (e.g., the existence of oblivious transfer), and the security guarantees that can be achieved.

Since in the (potentially) dishonest-majority setting, which is the focus in this work, *fairness* (either all parties learn the output or nobody does) cannot be achieved generically [19], the standard security requirement is weakened by allowing the adversary to prematurely abort the computation even after learning the output value. Three main flavors of this definition—distinguished by the guarantees that honest parties receive upon abort—have been considered in the literature:

1. Security with *identifiable abort* [49, 20] allows the honest parties to identify cheating parties in case of an abort;
2. security with *unanimous abort* [39, 28] allows the honest parties to detect that an attack took place, but not to catch the culprits; and, finally,
3. security with *selective (non-unanimous) abort* [40, 48] guarantees that every honest party either obtains the correct output from the computation or locally detects an attack and aborts.

We note in passing that the above ordering reflects the strength of the security definition, i.e., if a protocol is secure with identifiable abort then it is also secure with unanimous abort; and if a protocol is secure with unanimous abort, then it is also secure with selective abort. The opposite is not true in general.

A common design principle for MPC protocols, used in the vast majority of works in the literature, is to consider a *broadcast channel* as an atomic resource of the communication model. The ability to broadcast messages greatly simplifies protocols secure against malicious parties (see, e.g., the discussion in Goldreich's book [38, Sec. 7]) and is known to be necessary for achieving security with identifiable abort [20]. Indeed, broadcast protocols that run over authenticated channels exist assuming a public-key infrastructure (PKI) for digital signatures [26], with information-theoretic variants in the private-channels setting [62]. Therefore, in terms of *feasibility results* for MPC, the broadcast resource is interchangeable with a PKI setup. In fact, if merely *unanimous abort* is required, even this setup assumption can be removed [28].[1]

However, as discussed above, in terms of *round efficiency*, removing the broadcast resource is not for free and one needs to either pay with more rounds to emulate broadcast [26, 29], or lessen the obtained security guarantees. However, very few generic ways to trade-off broadcast for weaker security have been proposed. A notable case is that of Goldwasser and Lindell [40], who showed how to compile any r-round MPC protocol π that is designed in the broadcast model into a $2r$-round MPC protocol over point-to-point channels at the cost of settling for the weakest security guarantee of *selective abort*, even if the original protocol π was secure with unanimous or identifiable abort. Interestingly, since as mentioned earlier broadcast protocols are expensive in terms of rounds and communication, most (if not all) practical implementations of MPC protocols use this compiler and therefore can only achieve selective abort [55, 56, 43, 53, 65, 44].

But even at this security cost, the compiler from Goldwasser and Lindell [40] does not achieve a round-preserving reduction as it induces a constant multiplicative blowup in the number of rounds. The reason is that, in a nutshell, this compiler has every broadcast round being emulated by a two-round echo multi-cast approach, where every party sends the message he intends to broadcast to all other parties, who then echo it to ensure that if two honest parties received inconsistent messages everyone can observe. Such a blowup is unacceptable when we are after protocols with the minimal round complexity of two rounds.

Two-round MPC protocols in the malicious setting were first explored in [36, 37], while recent years have witnessed exciting developments in two-round MPC [48, 5, 33, 30, 41, 50, 58, 16, 12, 31, 32, 10, 59, 35, 1, 63, 34, 3, 11, 2, 4, 25]. The current state of the art can be summarized as follows:

- Garg and Srinivasan [32] and Benhamouda and Lin [10] showed how to balance between the optimal round complexity and minimal cryptographic assumptions for MPC in the broadcast model, by showing that every function can be computed with unanimous abort using two broadcast rounds, assuming two-round oblivious transfer (OT) and tolerating $t < n$ corruptions.
- In the honest-majority setting, Ananth et al. [2] and Applebaum et al. [4] showed that security

[1] In some cases, the PKI assumption can be removed even for the strong notion of *guaranteed output delivery*, see [20, 22].

with selective abort can be achieved using two point-to-point rounds assuming OWF.

- Patra and Ravi [59] showed that in the plain model (without any setup assumptions, such as a PKI) security with unanimous abort cannot be achieved in two point-to-point rounds, and even if the first round can use a broadcast channel. As pointed out in [61], the lower-bounds proofs from [59] do not extend to a setting with private-coins setup.

While advancing our understanding of what kind of security can be achieved in two rounds, the picture derived from the results above is only partial and does not resolve the question of whether the feasibility results can be pushed further. For example, is it possible to obtain identifiable abort via two broadcast rounds for $t < n$? Is it possible to achieve selective abort via two point-to-point rounds for $t < n$? What security can be achieved when broadcast is used only in a single round in a two-round MPC protocol? This motivates the main question we study in this paper:

What is the tradeoff between the use of broadcast and achievable security in two-round MPC?

1.2 Our Contributions

We devise a complete characterization of the feasibility landscape of two-round MPC against arbitrarily many malicious corruptions, with respect to the above three levels of security (with abort) depending on availability of a broadcast channel. Specifically, we consider all possible combinations of broadcast and point-to-point rounds—where a point-to-point round consists of only point-to-point communication whereas in a broadcast round at least one party uses the broadcast channel—i.e., no broadcast round, one broadcast round, and two broadcast rounds.

Our results are summarized in Table 1. For simplicity we prove our positive results secure against a static t-adversary, for $t < n$. Although we do not see a specific reason why an adaptive adversary cannot be tolerated, treating this stronger case would need a careful modification of our arguments; we leave a formal treatment of an adaptive adversary as an open question. All our negative results hold for a static adversary, and hence also for an adaptive adversary, since the latter is a stronger adversary. We note that due to the ordering in strength of the security definitions discussed above, any positive (feasibility) result implies feasibility for any column to its left in the same row, and an impossibility result implies impossibility for any column to its right in the same row.

rounds		**security with abort**		
first	**second**	**selective**	**unanimous**	**identifiable**
BC	BC	✓	✓ GS [32],BL [10]	✓ **Cor 1.1** ([32, 10])
P2P	BC	✓	✓ **Thm 4.1**	✗ **Thm 3.9**
BC	P2P	✓	✗ **Thm 3.3**	✗
P2P	P2P	✓ **Thm 4.1**	✗ **Thm 3.3**	✗
BC	-	✗ HLP [42]	✗	✗

Table 1: Feasibility and infeasibility of two-round MPC facing a static, malicious $(n-1)$-adversary. Feasibility results hold assuming two-round OT in the CRS model. Impossibility results hold given any correlated randomness. A corollary with a citation of a paper should be interpreted as corollary of the results of the paper that was not explicitly stated in the paper.

Next, we give a more detailed description of the results and how they complement the current landscape.

Two broadcast rounds MPC. First, as a justification of our search for round-optimal protocols, we observe that as a straightforward corollary of Halevi et al. [42], we can exclude the existence of a single-round general MPC protocol—i.e., MPC for any function. This is true for any of the three security definitions, independently of whether or not the protocol uses a broadcast channel. We can thus focus our attention to protocols with two rounds.

Let us first consider the case where both rounds use a broadcast channel. A simple observation reveals that in this case the strongest notion of security with identifiable abort is feasible. Indeed, the recent results by Garg and Srinivasan [32] and Benhamouda and Lin [10] prove that assuming two-round OT, every function can be securely computed with unanimous abort, tolerating static, malicious corruptions of any subset of the parties.[2] A simple corollary shows that when starting with an inner protocol that is secure with identifiable abort (e.g., the GMW protocol [39]), the compiled protocol will also be secure with identifiable abort. The proof follows directly by inspecting either one of the proofs of [32, 10]. For completeness, we state this as a corollary below.

Corollary 1.1 ([32, 10]). *Assume the existence of a two-round OT protocol secure against a static malicious adversary in the CRS model and let $t < n$. Then, every efficiently computable n-party function can be securely computed with identifiable abort in the CRS model using two broadcast rounds tolerating a static malicious t-adversary.*

This leaves open the cases of both rounds being point-to-point rounds, and of one broadcast round and one point-to-point round, which constitute our main contributions. Interestingly, in the latter case the order of the rounds makes a difference on what security can be achieved.

1.2.1 Impossibility results

We start our investigation with proving the lower bounds illustrated in Table 1. Towards this goal, we describe a simple three-party function which, due to its properties, can be used in all the associated lower bounds. At a very high level, the chosen function f enjoys two core properties that will be crucial in our impossibility proofs: First, the function takes two inputs from a dedicated party, say P_3, but in any evaluation, the output depends on only one of these values (which of the two inputs is actually used is mandated by the input of the other two parties). Second, f has input independence with respect to P_1's input, i.e., an adversary corrupting P_2 and P_3 cannot bias their inputs depending on P_1's input. (See Section 3 for the function's definition.)

We note in passing that all our impossibility results hold assuming an arbitrary private-coin setup and are therefore not implied by any existing work. As a result, wherever in our statements broadcast is assumed for some round, the impossibility holds even if point-to-point channels are also available in this round. The reason is that as our proofs hold assuming an arbitrary private-coins setup (e.g, a PKI), the setup can be leveraged to implement secure point-to-point communication over broadcast (using encryption). Thus, adding point-to-point communication in a broadcast round cannot circumvent our impossibilities. This is not necessarily the case when no setup is allowed by the proof, which is an additional justification for proving impossibilities which hold even assuming setup.

Here is how we proceed in gradually more involved steps to complete the impossibility landscape: As a first, easy step we show, using the line of argumentation of HLP [42], that our function f is one of the functions which cannot be computed in a single round even against any one party being semi-honest. This excludes existence of single-round maliciously secure generic MPC protocol against dishonest majorities, even if the single round is a broadcast round, and even if we are settling

[2]In fact, [10] also requires NIZK, but this assumption can be removed (see [11]).

for security with selective abort and assume an arbitrary correlated-randomness setup (last row in Table 1).

Unanimous abort requires second round over broadcast. Next, we turn to two-round protocols and prove impossibility for securely computing f with unanimous abort when only the first round might use broadcast, i.e., the second round is exclusively over point-to-point (row 4 in Table 1). This implies that under this communication pattern, security with identifiable abort is also impossible. Looking ahead, this impossibility result is complemented by Theorem 4.1 (Item 2), which shows that security with selective abort can be achieved in this setting.

The proof is somewhat involved, although not uncommon in lower bounds, but can be summarized as follows: We assume, towards a contradiction, that a protocol π computing f with unanimous abort exists. We then look at an adversary corrupting P_1 and define a sequence of worlds in which P_1's second-round messages are gradually dropped—so that in the last world, (the adversarial) P_1 sends no messages to the other parties. By sequentially comparing neighboring worlds, we prove that in all of them, the parties cannot abort and they have to output the output of the function evaluated on the original inputs that were given to the parties. However, as in the last scenario P_1 sends no message in the second round, this means that P_2 and P_3 can compute the output (which incorporates P_1's input) already in the first round. This enables a rushing adversary corrupting P_2 and P_3 to evaluate $f(x_1, x_2, x_3)$ on his favorite inputs for x_2 and x_3 before even sending any protocol message, and depending on the output y decide whether he wants to continue playing with those inputs—and induce the output $y = f(x_1, x_2, x_3)$ on P_1—or change his choice of inputs to some x'_2 and x'_3 and induce the output $y' = f(x_1, x'_2, x'_3)$ on P_1. This contradicts the second property of f, i.e., input independence with respect to P_1's input against corrupted P_2 and P_3.

We note in passing that a corollary of [59, Thm. 5] (explicitly stated in the full version [60, Cor. 1]) excluded security with unanimous abort for the case of honest majorities, but only for protocols that are defined in the plain model, without any trusted setup assumptions. Indeed, as pointed out by the authors in [61], their proof technique does not extend to the setting with private-coin setup. In more detail, and to illustrate the difference, consider the setting where the first round is over broadcast (and possibly point-to-point channels) and the second is over point-to-point. The argument for ruling out unanimous abort in [60, Cor. 1] crucially relies on P_3 not be able to distinguish between the case where P_2 does not send messages to P_1 (over a private channel) and the case where P_1 claims not to receive any message. However, given a PKI and a CRS for NIZK, the private channel can be emulated over the broadcast message, and the sender can prove honest behaviour. In this case, P_3 can detect the event where P_2 is cheating towards P_1 in the first round; hence, P_1 and P_3 can jointly detect the attack.

Identifiable abort requires two broadcast rounds. As a final step, we consider the case where only the second round might use broadcast—i.e., the first round is over a point-to-point channel. In this case we prove that security with identifiable abort is impossible (row 2 in Table 1). This result, which constitutes the core technical contribution of our work, is once again, complemented by a positive result which shows how to obtain unanimous abort with this communication pattern (Theorem 4.1). The idea of the impossibility proof is as follows: Once again we start with an assumed protocol π (towards contradiction) and compare two scenarios, where the adversary corrupts P_1 in the first and P_2 in the second. The adversary lets the corrupted party run π, but drops any message exchanged between P_1 and P_2 in the first (point-to-point) round. By comparing the views on the two scenarios we show that aborting is not an option. Intuitively, the reason is that identifiable abort requires the parties to agree on the identity of a corrupted party; but the transcript of the two executions are identical despite the corrupted party's identity being different, which means that if the parties try to identify a cheater, they will get it wrong (with noticeable probability) in one of the two scenarios.

Subsequently, we compare the world where P_2 is corrupted with one where the adversary corrupts also P_1 but has him play honestly; the correctness of the protocol (and the fact that the protocol machines are not aware of who is corrupted) ensures that despite the fact that P_1 is corrupted, his initial input will be used for computing the output of the honest party (which recall cannot abort as its view is identical to the other two scenarios). Since P_2 sends nothing to P_3 in Round 1, the information upon which P_3 bases all the messages he sends has to be independent of P_2's messages. However, this will allow a rushing adversary corrupting P_1 and P_2 to learn all messages from P_3 before sending to P_3 any message, and use them to compute the function on different inputs for P_2 in order to learn both inputs of P_3, thereby violating the first property of the function discussed above.

Note that the above is only a sketch of the argument, and the formal proof needs to take care of a number of issues: First, since P_2 is corrupted, it is not clear that the adversary can have strategies which yield both inputs of P_3. Indeed, it could be that under the described attack scenarios, there is only one possible effective input for corrupted P_2, which would exclude the possibility of the above attack. We prove that this is not the case, and that using the honest strategy, the adversary can induce an execution in which the different input distributions required by the proofs are used in the evaluation of the function. Second, in order to extract the two inputs, the adversary needs to know the output as well as the effective corrupted inputs on which the function is evaluated under our above attack scenarios. We ensure this by a simple syntactic manipulation of the function, i.e., by requiring each party to locally (and privately) output its own input as used in the evaluation of the function's output.

Observe that although our results are proved for three parties, they can be easily extended to n parties by a standard player-simulation argument [45]—in fact, because our adversary corrupts 2 out of the 3 parties, our result exclude any adversary corrupting $t \geq 2n/3$ of the parties.

1.2.2 Feasibility results

Next, we proceed to provide matching upper bounds, showing that security with *unanimous* abort is feasible when the second round is over broadcast (even if the first round is over point-to-point), and that security with *selective* abort can be achieved when both rounds are over point-to-point channels. Our results are based on the compiler of Ananth et al. [2], who focused on information-theoretic security of two-round MPC in the honest-majority setting. Ananth et al. [2], initially adjusted the two-round protocol from [1] to provide information-theoretic security with unanimous abort in the broadcast model (for NC^1 circuits), and then compiled it to provide security with selective abort over point-to-point channels.[3]

Compiling two-broadcast-round protocols. We start by presenting an adaptation of the compiler from [2] to the dishonest-majority setting. Let π_{bc} be a two-round MPC protocol in the broadcast model that is secure with unanimous abort. We first discuss how to compile π_{bc} to a protocol in which the first round is over point-to-point and the second round is over broadcast.

– In the compiled protocol, every party P_i starts by computing its first-round message in π_{bc}, denoted m_i^1. In addition, P_i considers its next-message function for the second round $\mathsf{second\text{-}msg}_i(x_i, r_i, m_1^1, \ldots, m_n^1)$ (that computes P_i's second round message based on its input

[3]We note that the approach of Applebaum et al. [4] does not extend to the dishonest-majority setting in a straightforward way.

x_i, randomness r_i, and all first-round messages). Each party "hard-wires" its input and randomness to the circuit computing $\mathsf{second\text{-}msg}_i$ such that given all first-round messages as input, the circuit outputs P_i's second-round message. Next, P_i garbles this circuit and secret-shares each input label using an additive secret-sharing scheme. In the first round of the compiled protocol, each party sends to each other party over private channels his first-round message from π_{bc} and one share of each garbled label. (Note that for all the parties, the "adjusted" second-round circuits should receive the same input values, i.e., the first-broadcast-round messages.)

- In case P_i didn't receive messages from all other parties he aborts. Otherwise, P_i receives from every P_j the message $m^1_{j \to i}$ (i.e., first-round messages of π_{bc}) and for each input wire of the next-message function of P_j, two shares: one for value 0 and the other for value 1 (recall that each bit that is broadcasted in the first round of π_{bc} forms an input wire in each circuit). In the second round, every party sends to all other parties the garbled circuit as well as one share from each pair, according to the messages received in the first round $(m^1_{1 \to i}, \ldots, m^1_{n \to i})$.
- Next, every party reconstructs all garbled labels and evaluates each garbled circuit to obtain the second-round messages of π_{bc}. Using these messages the output value from π_{bc} is obtained.

Proof intuition. Intuitively, if all honest parties receive the same "common part" of the first-round message (corresponding to the first broadcast round of π_{bc}), they will be able to reconstruct the garbled labels and obtain the second-round message of each party by evaluating the garbled circuits. Note that since the second round is over broadcast, it is guaranteed that all honest parties will evaluate the same garbled circuits using the same garbled inputs, and will obtain the same output value. If there exists a pair of parties that received different first-round messages, then none of the parties will be able to reconstruct the correct labels.

Given an adversary $\mathcal{A}_{\mathsf{out}}$ to the outer protocol (that uses a first point-to-point round) a simulator $\mathcal{S}_{\mathsf{out}}$ is constructed using a simulator $\mathcal{S}_{\mathsf{in}}$ for the inner protocol (in the broadcast model). At a high level, $\mathcal{S}_{\mathsf{out}}$ will use $\mathcal{S}_{\mathsf{in}}$ to simulate the first-round messages of the honest parties, send them (with the appropriate synthetic adjustments) to $\mathcal{A}_{\mathsf{out}}$, and get the corrupted parties' first-round messages.

- In case they are not consistent, $\mathcal{S}_{\mathsf{out}}$ will send abort to the trusted party and resume by simulating garbled circuits that output dummy values in the second round—this is secure since the labels for these garbled circuits will not be revealed.
- In case they are consistent, $\mathcal{S}_{\mathsf{out}}$ will use the inner simulator $\mathcal{S}_{\mathsf{in}}$ to extract the input values of the corrupted parties and send them to the trusted party. Once receiving the output, $\mathcal{S}_{\mathsf{out}}$ can hand it to $\mathcal{S}_{\mathsf{in}}$ who outputs the second-round messages for the honest parties. Next, $\mathcal{S}_{\mathsf{out}}$ will use these messages to simulate the garbled circuits of the honest parties and hand them to $\mathcal{A}_{\mathsf{out}}$. Based on the response from $\mathcal{A}_{\mathsf{out}}$ (i.e., the second-round messages) $\mathcal{S}_{\mathsf{out}}$ will send abort or $\mathsf{continue}$ to the trusted party and halt.

We remark that the proof in [2] also follows this intuition; however, that proof uses specific properties of the (simulator for the) broadcast-model protocol constructed in [2] (which in turn is based on the protocol from [1]). Our goal is to provide a generic compiler, which works for *any* two-round broadcast-model protocol, and so our use of the simulator for the broadcast-model protocol must be black-box. For that purpose, we devise non-trivial new simulation techniques, which we believe might be of independent interest. Our proof can be adapted to demonstrate that the original compilation technique of [2] is, in fact, generic, i.e., can securely compile any broadcast-hybrid protocol.

To explain the technical challenge and our solution, let us discuss the above issue in more detail: Recall that the security definition for the stand-alone model[4] from [38] guarantees that for every

[4]Our choice to describe the results in the stand-alone model is for simplicity and for providing stronger impos-

adversary there is a simulator for the ideal computation (in the current case, ideal computation with unanimous abort). The simulator is invoked with some auxiliary information, and starts by sending to the trusted party inputs for the corrupted parties (or abort). Upon receiving the output value, the simulator responds with $\mathsf{abort}/\mathsf{continue}$, and finally generates its output which is computationally indistinguishable from the view of the adversary in a protocol (where the honest parties' outputs are distributed according to the extracted corrupted-parties' inputs).

Given an adversary $\mathcal{A}_{\mathsf{out}}$ for the compiled protocol π, we would like to use the security of π_{bc} to construct a simulator $\mathcal{S}_{\mathsf{out}}$ and simulate the "common part" of the honest parties' messages (i.e., the messages $m^1_{i\to j}$ from an honest P_i to a corrupted P_j). However, the adversary $\mathcal{A}_{\mathsf{out}}$ induces *multiple adversaries* for π_{bc}, one for every honest party and it is not clear which simulator (i.e., for which of these adversaries) should be used. In fact, before interacting with $\mathcal{A}_{\mathsf{out}}$ and sending him the first-round messages of honest parties, $\mathcal{S}_{\mathsf{out}}$ should first run one (or a few) of the aforementioned simulators to get the inputs for the corrupted parties, invoke the trusted party with the input values, and get back the output. (At this point the simulator is committed to the corrupted parties' inputs.)[5] Only then can $\mathcal{S}_{\mathsf{out}}$ send the output back to the inner simulator(s) and get the view of the inner adversary (adversaries) in the execution, and use it to interact with $\mathcal{A}_{\mathsf{out}}$.

Receiver-specific adversaries. To solve this conundrum, we construct our simulator as follows: For every honest party P_j we define a *receiver-specific adversary* $\mathcal{A}^j_{\mathsf{in}}$ for π_{bc}, by forwarding the first-broadcast-round messages to $\mathcal{A}_{\mathsf{out}}$ and responding with the messages $\mathcal{A}_{\mathsf{out}}$ sends to P_j (recall that $\mathcal{A}_{\mathsf{out}}$ can send different messages to different honest parties in π). By the security of π_{bc}, for every such $\mathcal{A}^j_{\mathsf{in}}$ there exists a simulator $\mathcal{S}^j_{\mathsf{in}}$.

To define the simulator $\mathcal{S}_{\mathsf{out}}$ (for the adversary $\mathcal{A}_{\mathsf{out}}$), we use one of the simulators $\mathcal{S}^j_{\mathsf{in}}$ corresponding to the honest parties. $\mathcal{S}_{\mathsf{out}}$ initially receives from $\mathcal{S}^j_{\mathsf{in}}$ either the corrupted parties' inputs or an abort message, and forwards the received message to the trusted party. If $\mathcal{S}^j_{\mathsf{in}}$ does not abort, $\mathcal{S}_{\mathsf{out}}$ receives back the output value y, forwards y to $\mathcal{S}^j_{\mathsf{in}}$ and receives the simulated second-round messages from $\mathcal{S}^j_{\mathsf{in}}$'s output. Next, $\mathcal{S}_{\mathsf{out}}$ invokes $\mathcal{A}_{\mathsf{out}}$ and simulates the first-round messages of π (using the simulated first-round messages for π_{bc} obtained from $\mathcal{S}^j_{\mathsf{in}}$), receives back the first-round messages from $\mathcal{A}_{\mathsf{out}}$, and checks whether these messages are consistent. If so, $\mathcal{S}_{\mathsf{out}}$ completes the simulation by constructing simulated garbled circuits that output the correct second-round messages (if $\mathcal{A}_{\mathsf{out}}$'s messages are consistent, the simulated messages by $\mathcal{S}^j_{\mathsf{in}}$ are valid for all honest parties). If $\mathcal{A}_{\mathsf{out}}$'s messages are inconsistent, $\mathcal{S}_{\mathsf{out}}$ simulates garbled circuits that output dummy values (e.g., zeros), which is acceptable since the $\mathcal{A}_{\mathsf{out}}$ will not learn the labels to open them. We refer the reader to Section 4.2 for a detailed discussion and a formal proof.

Selective abort via two point-to-point rounds. After showing that the compiler from [2] can be adjusted to achieve unanimous abort when the first round is over point-to-point and the second is over broadcast, we proceed to achieve selective abort when both rounds are over point-to-point, facing any number of corruptions. The main difference from the previous case is that the adversary can send different garbled circuits to different honest parties in the second round, potentially causing them to obtain different output values, which would violate correctness (recall that the definition of security with selective abort permits some honest parties to abort while other obtain the correct output, but it is forbidden for two honest parties to obtain two different output

sibility results. Our feasibility results extend to the UC framework [14] via standard technical adjustments, as our simulators are black-box and straight-line. We note that the same simulation techniques discussed in this section are also needed for adjusting the proof to the UC model.

[5]This is challenging because we use the broadcast-hybrid protocol in a black-box manner. Restricting to subclasses of protocols with specific properties—e.g., the view of the adversary in the first round is distributed independently of the function's output—may enable more straightforward simulation strategies.

values). However, we reduce this attack to the security of π_{bc} and show that it can only succeed with negligible probability. See Section 4.3 for details.

Organization of the paper. Preliminaries are presented in Section 2. In Section 3 we present our impossibility results and in Section 4 our feasibility results.

2 Preliminaries

In this section, we introduce some necessary notation and terminology. We denote by κ the security parameter. For $n \in \mathbb{N}$, let $[n] = \{1, \cdots, n\}$. Let poly denote the set of all positive polynomials and let PPT denote a probabilistic algorithm that runs in *strictly* polynomial time. A function $\nu \colon \mathbb{N} \to [0,1]$ is *negligible* if $\nu(\kappa) < 1/p(\kappa)$ for every $p \in \mathsf{poly}$ and large enough κ. Given a random variable X, we write $x \leftarrow X$ to indicate that x is selected according to X.

2.1 Garbling Schemes

We now give a formal definition of a garbling scheme [66, 7].

Definition 2.1. *A projective* garbling scheme *is a pair* $\Pi = (\mathsf{Garble}, \mathsf{Eval})$ *such that:*

- $(\mathsf{GC}, \boldsymbol{K}) \leftarrow \mathsf{Garble}(1^\kappa, C)$*: on input the security parameter* 1^κ *and a Boolean circuit* $C : \{0,1\}^\ell \to \{0,1\}^m$*, the* garbling *algorithm outputs a garbled circuit* GC *and* ℓ *pairs of garbled labels* $\boldsymbol{K} = (K_1^0, K_1^1, \ldots, K_\ell^0, K_\ell^1)$*. For simplicity, we assume that for every* $i \in [\ell]$ *and* $b \in \{0,1\}$*, it holds that* $K_i^b \in \{0,1\}^\kappa$*.*
- $y = \mathsf{Eval}(\mathsf{GC}, K_1, \ldots, K_\ell)$*: on input a garbled circuit* GC *and* ℓ *garbled labels* $K_1, \ldots, K_\ell$ *the* evaluation *algorithm outputs a value* $y \in \{0,1\}^m$*.*

We require the following properties from a garbling scheme:

Correctness. For any Boolean circuit $C : \{0,1\}^\ell \to \{0,1\}^m$ and $x = (x_1, \ldots, x_\ell) \in \{0,1\}^\ell$ it holds that

$$\Pr\left[\mathsf{Eval}(\mathsf{GC}, \boldsymbol{K}[x]) \neq C(x)\right] = \mathsf{negl}(\kappa),$$

where $(\mathsf{GC}, \boldsymbol{K}) \leftarrow \mathsf{Garble}(1^\kappa, C)$ with $\boldsymbol{K} = (K_1^0, K_1^1, \ldots, K_\ell^0, K_\ell^1)$, and $\boldsymbol{K}[x] = (K_1^{x_1}, \ldots, K_\ell^{x_\ell})$.

Privacy. There exists a PPT simulator SimGC such that for every PPT adversary $\mathcal{A}$

$$\left|\Pr\left[\mathsf{Expt}^{\mathsf{garble}}_{\Pi,\mathcal{A},\mathsf{SimGC}}(\kappa, 0) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{garble}}_{\Pi,\mathcal{A},\mathsf{SimGC}}(\kappa, 1) = 1\right]\right| \leq \mathsf{negl}(\kappa),$$

where the experiment $\mathsf{Expt}^{\mathsf{garble}}_{\Pi,\mathcal{A},\mathsf{SimGC}}(\kappa, b)$ is defined as follows:

1. The adversary $\mathcal{A}$ specifies $C : \{0,1\}^\ell \to \{0,1\}^m$ and $x = (x_1, \ldots, x_\ell) \in \{0,1\}^\ell$.
2. The challenger responds as follows:
   - If $b = 0$ set $(\mathsf{GC}, \boldsymbol{K}) \leftarrow \mathsf{Garble}(1^\kappa, C)$ with $\boldsymbol{K} = (K_1^0, K_1^1, \ldots, K_\ell^0, K_\ell^1)$. Responds with $(\mathsf{GC}, \boldsymbol{K}[x])$, where $\boldsymbol{K}[x] = (K_1^{x_1}, \ldots, K_\ell^{x_\ell})$.
   - If $b = 1$, respond with $(\mathsf{GC}, K_1, \ldots, K_\ell) \leftarrow \mathsf{SimGC}(1^\kappa, C, C(x))$.
3. The adversary outputs a bit b', which is the output of the experiment.

2.2 Security Model

We provide the basic definitions for secure multiparty computation according to the real/ideal paradigm (see [38, 13, 14] for further details), capturing in particular the various types of unsuccessful termination ("abort") that may occur. For simplicity, we state our results in the stand-alone setting, however, all of our results can be extended to the UC framework [14].

Real-world execution. An n-party protocol $\pi = (P_1, \ldots, P_n)$ is an n-tuple of PPT interactive Turing machines. The term *party* P_i refers to the i'th interactive Turing machine. 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* $\mathcal{A}$ 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 an additional auxiliary input. The parties execute the protocol in a synchronous network. That is, the execution proceeds in rounds: Each round consists of a *send phase* (where parties send their messages from this round) followed by a *receive phase* (where they receive messages from other parties). The adversary is assumed to be *rushing*, which means that he can see the messages the honest parties send in a round before determining the messages that the corrupted parties send in that round.

The parties can communicate in every round over a broadcast channel or using a fully connected point-to-point network. The communication lines between the parties are assumed to be ideally authenticated and private (and thus the adversary cannot modify messages sent between two honest parties nor read them).[6]

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 actively malicious, meaning that he 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 do not output anything and the adversary outputs an (arbitrary) function of its view of the computation (containing the views of the corrupted parties). The view of a party in a given execution of the protocol consists of its input, its random coins, and the messages it sees throughout this execution.

Definition 2.2 (Real-world execution)**.** *Let $\pi = (P_1, \ldots, P_n)$ be an n-party protocol and let $\mathcal{I} \subseteq [n]$ denote the set of indices of the parties corrupted by $\mathcal{A}$. The* joint execution of π under $(\mathcal{A}, \mathcal{I})$ in the real model*, on input vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, auxiliary input* aux *and security parameter κ, denoted $\textsc{real}_{\pi,\mathcal{I},\mathcal{A}(\mathsf{aux})}(\boldsymbol{x}, \kappa)$, is defined as the output vector of $P_1, \ldots, P_n$ and $\mathcal{A}(\mathsf{aux})$ resulting from the protocol interaction.*

Ideal-world execution (with abort). We now present standard definitions of ideal computations that are used to define security with identifiable abort, unanimous abort, and selective (non-unanimous) abort. For further details see [40, 49, 20].

An ideal computation with abort of an n-party functionality f on input $\boldsymbol{x} = (x_1, \ldots, x_n)$ for parties $(P_1, \ldots, P_n)$ in the presence of an adversary (a simulator) $\mathcal{S}$ controlling the parties indexed by $\mathcal{I} \subseteq [n]$, proceeds via the following steps.

Sending inputs to trusted party: An honest party P_i sends its input x_i to the trusted party. The adversary may send to the trusted party arbitrary inputs for the corrupted parties. Let x_i' be the value actually sent as the input of party P_i.

[6]Private channels can be realized over authenticated channels without increasing the round complexity given a PKI for public-key encryption.

Trusted party answers adversary: The trusted party computes $y = f(x'_1, \ldots, x'_n)$. If there are corrupted parties, i.e., if $\mathcal{I} \neq \emptyset$, send y to $\mathcal{S}$. Otherwise, proceed to step *Trusted party answers remaining parties.*

Adversary responds to trusted party: The adversary $\mathcal{S}$ can either select a set of parties that will not get the output by sending an $(\mathsf{abort}, \mathcal{J})$ message with $\mathcal{J} \subseteq [n] \setminus \mathcal{I}$, or allow all honest parties to obtain the output by sending a $\mathsf{continue}$ message.

Trusted party answers remaining parties: If $\mathcal{S}$ has sent an $(\mathsf{abort}, \mathcal{J})$ message with $\mathcal{J} \subseteq [n] \setminus \mathcal{I}$ and $\mathcal{I} \neq \emptyset$, the trusted party sends $\perp$ to every party P_j with $j \in \mathcal{J}$ and y to every P_j with $j \notin \mathcal{J} \cup \mathcal{I}$. Otherwise, if the adversary sends a $\mathsf{continue}$ message or if $\mathcal{I} = \emptyset$, the trusted party sends y to P_i for every $i \notin \mathcal{I}$.

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

Definition 2.3 **(Ideal computation with *selective* abort).** *Let $f\colon (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be an n-party functionality and let $\mathcal{I} \subseteq [n]$ be the set of indices of the corrupted parties. Then, the* joint execution of f under $(\mathcal{S}, \mathcal{I})$ in the ideal computation*, on input vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, auxiliary input* aux *to $\mathcal{S}$ and security parameter κ, denoted $\text{IDEAL}^{\mathsf{sl\text{-}abort}}_{f,\mathcal{I},\mathcal{S}(\mathsf{aux})}(\boldsymbol{x}, \kappa)$, is defined as the output vector of $P_1, \ldots, P_n$ and $\mathcal{S}$ resulting from the above described ideal process.*

We now define the following variants of this ideal computation:

- **Ideal computation with *unanimous* abort**. This ideal computation proceeds as in Definition 2.3, with the difference that in order to abort the computation, the adversary simply sends abort to the trusted party (without specifying a set $\mathcal{J}$). In this case, the trusted party responds with $\perp$ to all honest parties. This ideal computation is denoted as $\text{IDEAL}^{\mathsf{un\text{-}abort}}_{f,\mathcal{I},\mathcal{S}(\mathsf{aux})}(\boldsymbol{x}, \kappa)$.
- **Ideal computation with *identifiable* abort**. This ideal computation proceeds as the ideal computation with unanimous abort, with the exception that in order to abort the computation, the adversary chooses an index of a corrupted party $i^* \in \mathcal{I}$ and sends (abort, i^*) to the trusted party. In this case, the trusted party responds with $(\perp, i^*)$ to all parties. This ideal computation is denoted as $\text{IDEAL}^{\mathsf{id\text{-}abort}}_{f,\mathcal{I},\mathcal{S}(\mathsf{aux})}(\boldsymbol{x}, \kappa)$.

Security definitions. Having defined the real and ideal computations, we can now define security of protocols.

Definition 2.4. *Let* $\mathsf{type} \in \{\mathsf{sl\text{-}abort}, \mathsf{un\text{-}abort}, \mathsf{id\text{-}abort}\}$. *Let $f\colon (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be an n-party functionality. A protocol π t-*securely computes* f *with* "type" *if for every PPT real-world adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{S}$, such that for every $\mathcal{I} \subseteq [n]$ of size at most t, it holds that*

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

3 Impossibility Results

In this section, we prove out impossibility results. Concretely, in Section 3.1, we argue that there is no single-round maliciously secure generic MPC protocol against dishonest majorities, even if the single round is a broadcast round, and even if we are settling for security with selective abort and we assume an arbitrary correlated-randomness setup. Subsequently, in Section 3.2, we prove that no generic two-round MPC protocol can achieve security with identifiable abort, while making use of broadcast in only one of the two rounds. This holds irrespective of whether the broadcast round is the first or second one. Towards this goal, we start by proving that no two-round protocol in which the broadcast round is first—i.e., the second round is over point-to-point—can achieve *identifiable* abort. This is proved in Theorem 3.3; in fact, the theorem proves a stronger statement, namely, that there is a function f such that no protocol with the above structure can securely compute f with *unanimous* abort.[7]

Theorem 3.3 implies that the only option for a two-round protocol with only one broadcast round to securely compute f with identifiable abort, is if the broadcast round is the second round—i.e., the first round is over point-to-point. We prove (Theorem 3.9) that this is also impossible, i.e., f cannot be computed by such a protocol. This proves that the result from Theorem 4.1 (Item 1), which achieves security with unanimous abort in this case, is also tight and completes the (in)feasibility landscape for two-round protocols. Furthermore, we note that all the results proved in this section hold for both computational and information-theoretic security, even if we assume access to an arbitrary correlated-randomness setup.

A simple function. Before starting our sequence of impossibility results, we first introduce a simple function which we will use throughout this section. Consider the following three-party public-output function (i.e., all three parties receive the output): The parties, P_1, P_2, and P_3, hold inputs $x_1 \in \{0,1\} \times \{0,1\}$, $x_2 \in \{0,1\}$ and $x_3 \in \{0,1\}^k \times \{0,1\}^k$, respectively, where $x_1 = (x_{1,1}, x_{1,2})$ and $x_3 = (x_{3,1}, x_{3,2})$. For a bit b we denote by b^κ the string resulting from concatenating κ times the bit b (recall that κ denotes the security parameter). The function is defined as follows:

$$f(x_1, x_2, x_3) = \begin{cases} x_{1,1}^\kappa \oplus x_2^\kappa \oplus x_{3,1}, \text{ if } x_{1,2} = x_2 \\ x_{1,1}^\kappa \oplus x_2^\kappa \oplus x_{3,2}, \text{ if } x_{1,2} \neq x_2. \end{cases}$$

Note that in the above function, the first bit of P_1, i.e., $x_{1,1}$ contributes to the computed XOR, whereas the relation between the second bit of P_1, i.e., $x_{1,2}$, and the input-bit x_2 of P_2 is the one which defines which of the $x_{3,1}$ or $x_{3,2}$ will be used in the output. One can easily verify that the following is a more compact representation of f:

$$f(x_1, x_2, x_3) = x_{1,1}^\kappa \oplus x_2^\kappa \oplus x_{3,1+(x_{1,2} \oplus x_2)}.$$

The latter representation will be useful in the proof of Theorem 3.9.

As discussed in the introduction, the above function enjoys the following two useful properties: First, it is impossible in the ideal world (where parties and an adversary/simulator have access to a TTP for f) for the simulator to learn both inputs of P_3 even if he corrupts both P_1 and P_2. Second, assuming the input $x_{1,1}$ of P_1 is chosen uniformly at random, it is impossible for a simulator corrupting P_2 and P_3 to fix the output to 0. We prove these two properties in the corresponding theorems where they are used.

3.1 Impossibility of Single-Round MPC

As a simple corollary of HLP [42] (see also [59]), we can exclude the existence of a semi-honestly secure MPC protocol for the above function. For completeness, we sketch the proof in Appendix A.

[7]Recall that there is a trivial reduction from security with unanimous abort to security with identifiable abort: Run the protocol and in case it aborts with the ID of some party P_i, output abort and ignore the identity of the corrupted party.

Corollary 3.1 ([42])**.** *The function f cannot be computed with selective abort by a single-round protocol tolerating one semi-honest corrupted party.*

Extending Corollary 3.1 to the multi-party case (involving more than three parties) follows using a player-simulation argument, and the following facts that are implied by our definition of security with selective abort: (1) If the adversary follows his protocol, the evaluation cannot abort even if parties are corrupted; this follows from the non-triviality condition and the fact that when the adversary follows the protocol with his corrupted parties, the protocol cannot deviate based on the fact that parties are corrupted; (2) for such an honest-looking adversary [15], the protocol achieves all the guarantees required for semi-honest security—i.e., there is a simulator which simulates the adversary's entire view from the inputs and outputs of corrupted parties. We refer to Appendix A for a proof.

Corollary 3.2. *For $n \geq 3$, there exist an n-party function f_n for which there is no single-round protocol π which securely computes f_n with selective abort against even a single corruption. The statement is true even if π uses a broadcast channel in its single round.*

3.2 Impossibility of Single-Broadcast Two-Round MPC

Having excluded the possibility of single-round MPC protocols, we next turn to two rounds. Throughout this section, we prove impossibility statements for three-party protocols (for the function f). As discussed in the introduction, all our statements can be directly extended to the multi-party setting using the straightforward extension of f to n parties (cf. function f_n in Corollary 3.2).

3.2.1 Impossibility of Unanimous Abort when Broadcast is First Round

We start by proving impossibility of security with unanimous abort for f against corrupted majorities. Analogous to [42] we will say that *an adversary learns the residual function* $f(x_1, \cdot, \cdot)$ to denote the event that the adversary learns enough information to locally and efficiently compute $f(x_1, x_2^*, x_3^*)$ on any (and as many) inputs x_2^* and x_3^* as he wants.

Theorem 3.3. *There exists no two-round protocol π which securely computes f with unanimous abort against corrupted majorities while making use of the broadcast channel only in the first round (i.e., where the second round is over point-to-point channels). The statement is true even assuming an arbitrary correlated randomness setup.*

Proof. Towards a contradiction, assume that there is protocol $\pi = (\pi_1, \pi_2, \pi_3)$, where π_i is the code (e.g., interactive Turing machine) of P_i, for computing f with unanimous abort which uses broadcast in its first round, but only point-to-point in the second round. Consider executions of π on uniformly random inputs x_1 and x_2 for P_1 and P_2 and on input $x_3 \in \{(0^\kappa, 1^\kappa), (1^\kappa, 0^\kappa)\}$ from P_3 in the following scenarios (see Figure 1 for an illustration). In all four scenarios, the adversary uses the honest input for the corrupted party and allows him to execute his honest protocol on uniform random coins, but might drop some of the messages the corrupted party's protocol attempts to send in Round 2.

Scenario 1: The adversary corrupts P_1, plays the first round according to π but sends no messages in the second round.

Scenario 2: The adversary corrupts P_1, plays both rounds according to π, but does not sent his second-round message towards P_3; party P_2 receives his second-round message according to the honest protocol.

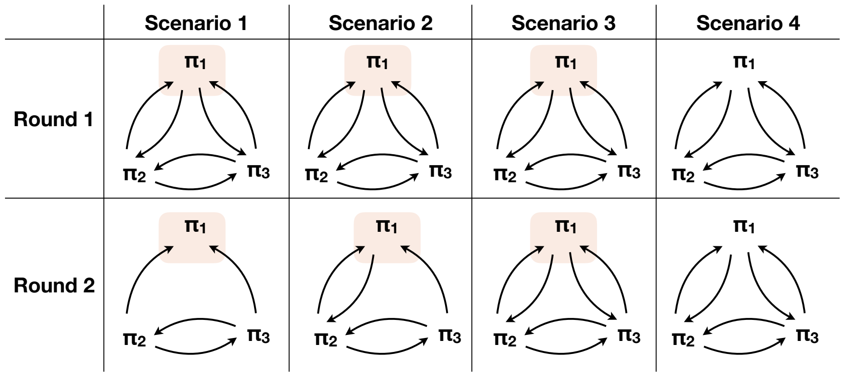

Figure 1: The scenarios from the proof. All protocols are executed as specified; whenever an arrow is present it indicates that the message that the corresponding protocol would send is indeed sent; missing arrows indicate that respective messages are dropped. A shade on the background of a protocol indicates that the corresponding party is corrupted (but the adversary still executes the respective protocol on the honest input, but might drop some messages).

Scenario 3: The adversary corrupts P_1 but plays the honest protocols in both rounds.

Scenario 4: No party is corrupted.

The proof of the theorem proceeds as follows: By a sequence of comparisons between the four scenarios we show that in Scenario 1, π_2 and π_3 cannot abort and will have to produce output equal to $f(x_1, x_2, x_3)$ with overwhelming probability despite the fact that P_1 sends no message in Round 2. This means that a (rushing)[8] adversary corrupting P_2 can learn the residual function $f(x_1, \cdot, \cdot)$ already in Round 1 and before committing to any inputs for P_2 and P_3. This allows him to choose corrupted inputs depending on (the honest input) x_1 violating the security (in particular the input-independence property)[9] of π. The formal argument follows. For notational clarity, we will denote the message that P_i sends to P_j over a point to point channel in round ρ by $m_{\rho,i\to j}$; if in round ρ a party P_i broadcasts a messages, we will denote this message by $m_{\rho,i\to *}$.

Claim 3.4. *In Scenario 3, parties P_2 and P_3 output $f(x_1, x_2, x_3)$ with overwhelming probability.*

Proof. The claim follows by the correctness of the protocol π. Indeed, the views of P_2 and P_3 in Scenario 3 are distributed identically to their views in Scenario 4. This means that on any inputs x_1, x_2, x_3 for protocols π_1, π_2, π_3, the output of P_2 and P_3 in Scenario 3 must be identically distributed to his output in Scenario 4; however, by the security (in particular the correctness) of π this output will be $f(x_1, x_2, x_3)$ with overwhelming probability. □

Claim 3.5. *In Scenario 2, parties P_2 and P_3 output $f(x_1, x_2, x_3)$ with overwhelming probability.*

Proof. The view of P_2 in Scenario 2 is identically distributed to his view in Scenario 3. As before, this means that his output in Scenario 2 must be identical to that in Scenario 3, which, as proved

[8]Our impossibility results consider standard, worst-case and rushing adversaries. One might investigate how the landscape looks like against non-rushing adversaries, but this is typically considered too strong an assumption for protocols, as it implies feasibility of fair exchange (a task impossible in the standard rushing-adversary with dishonest majority realm) and even in a single round. We do not consider this theoretical question here.

[9]Informally, input independence, a property implied by the standard simulation-based security definition (see Section 2.2), requires that the adversary cannot choose his inputs depending on the inputs of honest parties.

in the previous claim, equals $f(x_1, x_2, x_3)$ with overwhelming probability—in particular P_2 does not abort. However, by the unanimous abort requirement, the fact that (the honest) P_2 does not abort means that (the also honest) P_3 also does not abort except with negligible probability. But then, the correctness of π (recall that f has the same output for all honest parties) implies that with overwhelming probability the output of P_3 will also be $f(x_1, x_2, x_3)$. □

Claim 3.6. *In Scenario 1, parties P_2 and P_3 output $f(x_1, x_2, x_3)$ with overwhelming probability.*

Proof. The proof is similar to the above claim: The view of P_3 in Scenario 2 is identically distributed to his view in Scenario 1. As before, this means that on any inputs x_1, x_2, x_3 for π, P_3's output in Scenario 2 must be identical to that in Scenario 1, which, as proved in the previous claim, equals $f(x_1, x_2, x_3)$ with overwhelming probability—in particular P_3 does not abort. However, by the unanimous abort requirement, the fact that (the honest) P_3 does not abort means that P_2 also does not abort except with negligible probability. But then, the correctness of π (recall that f has the same output for all honest parties) implies that with overwhelming probability the output of P_2 will also be $f(x_1, x_2, x_3)$. □

Claim 3.7. *An adversary corrupting P_2 and P_3 can learn the residual function $f(x_1, \cdot, \cdot)$ before P_2 or P_3 send any message.*

Proof. Recall that, from the last claim, in Scenario 1, both P_2 and P_3 output $f(x_1, x_2, x_3)$ with overwhelming probability. However, since in this scenario P_1 does not send anything in Round 2, this implies that there exists an adversary $\mathcal{A}$ corrupting P_2 and P_3, who can with overwhelming probability already compute the residual function $f(x_1, \cdot, \cdot)$ using only the message $m_{1,1\to*}$ which P_1 broadcasts in round 1. Indeed, such an adversary needs to simply run the following strategy: receive $m_{1,1\to*}$ from P_1; simulate P_2 and P_3 on any inputs x_2^* and x_3^*; simulate the first-round messages $m_{1,2\to*}$ and $m_{1,3\to*}$ (i.e., the messages that π_2 and π_3 would broadcast if they were allowed to execute π); simulate the second-round messages $m_{2,2\to1}, m_{2,2\to3}, m_{2,3\to1}, m_{2,3\to2}$ of P_2 and P_3 on inputs, the (simulated) first-round messages, and the actual first-round messages received from P_1, and output their output. It follows from Claim 3.6 that this attack will output $f(x_1, x_2^*, x_3^*)$ except with negligible probability. Note that in this attack, $\mathcal{A}$ did not need to send any message to P_1. Furthermore, this attack can be repeated from the point of receiving $m_{1,1\to*}$ with any inputs for P_2 and P_3. □

To complete the proof of the theorem, we show that existence of the above adversary $\mathcal{A}$ implies an adversary $\mathcal{A}'$ that can break the security (in particular, the input independence) of π. Intuitively, $\mathcal{A}'$ will corrupt P_2 and P_3 and use the strategy of the adversary $\mathcal{A}$ from the above claim to learn the residual function before committing to his own input to f; thus $\mathcal{A}'$ is free to choose this inputs for P_2 and P_3 depending on x_1. We next provide a formal proof of this fact by describing a strategy for biasing the output (depending on x_1) which cannot be simulated

Concretely, consider the following $\mathcal{A}'$ that corrupts P_2 and P_3: $\mathcal{A}'$ receives $m_{1,1\to*}$ from P_1 and using $\mathcal{A}$, for $x_2^* = 0$ and $x_{3,1}* = 0^\kappa$ and $x_{3,2}* = 1^\kappa$, $\mathcal{A}'$ computes $y = f(x_1, 0, (0^\kappa, 1^\kappa))$. Then, dependent on whether y is 0^k or 1^k—observe that by definition of the function, these are the only two possible outcomes given the above inputs of P_3—$\mathcal{A}'$ distinguishes two cases:

Case 1: If $y = 0^\kappa$ then execute the honest protocol for P_2 and P_3 with these inputs, i.e., $x_2 = 0$ and $x_{3,1} = 0^\kappa$ and $x_{3,2} = 1^\kappa$.

Case 2: If $y = 1^\kappa$, then execute the honest protocol for P_2 and P_3 with the inputs of P_3 swapped, i.e., $x_2 = 0$ and $x_{3,1} = 1^\kappa$ and $x_{3,2} = 0^\kappa$.

Note that in both cases P_1 witnesses a view which is indistinguishable from the honest protocol with inputs: $x_2 = 0$ and $x_{3,1} = 0^\kappa$ and $x_{3,2} = 1^\kappa$ (Case 1) or $x_2 = 0$ and $x_{3,1} = 1^\kappa$ and $x_{3,2} = 0^\kappa$ (Case 2); hence, the correctness of π implies that with overwhelming probability if $y = f(x_1, 0, (0^\kappa, 1^\kappa)) = 0^\kappa$ then P_1 will output it, otherwise, i.e., if $y = f(x_1, 0, (0^\kappa, 1^\kappa)) = 1^\kappa$ he will output $y = f(x_1, 0, (1^\kappa, 0^\kappa))$; but in this latter case $y = 0^\kappa$ by the definition of f. Hence, this adversary always makes P_1 output 0^κ.

To complete the proof we prove that in an ideal evaluation of f with an honest P_1 and corrupted P_2 and P_3, if P_1 uses a uniformly random input and no abort occurs, then the output can be 0^κ with probability at most $1/2 \pm \mathsf{negl}(\kappa)$.

Claim 3.8. *For any simulator $\mathcal{S}$ corrupting P_2 and P_3 and not causing the ideal execution to abort, if P_1's input is chosen uniformly at random, then for any choice of inputs for P_2 and P_3, there exist a string $z \in \{0,1\}^\kappa$ such that the output of P_1 will be z or $\bar{z}$ each with probability $1/2 \pm \mathsf{negl}(\kappa)$.*

Proof. Let $X_1 = (X_{1,1}, X_{1,2}), X_2, X_3 = (X_{3,1}, X_{3,2})$ denote the random variables corresponding to the inputs of the parties, P_1, P_2, and P_3, respectively. Then, the output of f can be described as the random variable $Z = X_{1,1}^\kappa \oplus X_2^\kappa \oplus X_{3,1+(X_{1,2}\oplus X_2)}$. Since P_1 chooses his input uniformly at random, the distribution of $X_{1,1}$ is uniform and independent of the joint distribution of $(X_{1,2}, X_2, X_3)$. Hence, for any choice of input x_2 and x_3 of the simulator $\mathcal{S}$ for P_2 and P_3, and any choice of $x_{1,2}$ from P_1, the output of the function f will be either $z = x_2^\kappa \oplus x_{3,1+(x_{1,2}\oplus x_2)}$ if $x_{1,1} = 0$ or $z' = x_2^\kappa \oplus x_{3,1+(x_{1,2}\oplus x_2)} + 1^\kappa = \bar{z}$, if $x_{1,1} = 1$. But since $x_{1,1}$ is chosen uniformly and independently of any other input, each of z and $\bar{z}$ might occur with probability $1/2$. ☐

The above claim implies that for any simulator, with probability at least $1/2$ the output will be different than 0^κ. Hence the adversary $\mathcal{A}'$ (who, recall, always fixes the output to 0^κ) cannot be simulated which contradicts the assumed security of π. ☐

3.2.2 Impossibility of Identifiable Abort

Next, we proceed to the proof of our second, and main, impossibility theorem about identifiable abort. For this proof we make the following modification to f: In addition to its output from f, every party P_i is required to locally output his own input x_i. We denote this function by $\hat{f}$. Specifically, the output of $\hat{f}$ consists of two parts: A public part that is identical to f, which is the same for all parties (without loss of generality, we will use $f(x_1, x_2, x_3)$ to denote this part), and a private part which for each P_i is its own input.

$$\hat{f}(x_1, x_2, x_3) = ((y, x_1), (y, x_2), (y, x_3)) \quad \text{where} \quad y = f(x_1, x_2, x_3).$$

We remark that impossibility for such a public/private output function $\hat{f}$ implies impossibility of public output functions via the standard reduction of private to public input functions (see [38]).

Theorem 3.9. *The function $\hat{f}$ cannot be securely computed with identifiable abort by a three-party protocol that uses one point-to-point round and one broadcast round, tolerating (up to) two corrupted parties. This is true even assuming an arbitrary correlated-randomness setup.*

Proof. Assume, towards a contradiction, that a protocol π exists for the function $\hat{f}$. First, note that due to Theorem 3.3, the broadcast round cannot be the first round. (This holds because security with identifiable abort implies security with unanimous abort.) Hence, the first round of π must be the point-to-point round and the second can be a broadcast round. In the following, we will assume that the second round uses only the broadcast channel; this is without loss of generality as we allow π to be in the correlated-randomness model, which means that parties might share keys that they can use to emulate point-to-point communication over the broadcast network. (Proving impossibility in the correlated-randomness model implies impossibility in the plain model.)

Consider the parties P_1, P_2, and P_3 holding uniformly chosen inputs x_1, x_2, and x_3 for $\hat{f}$. Let π_i denote the code executed by P_i in π (i.e., P_i's protocol machine), and consider the following scenarios (also illustrated in Figure 2):

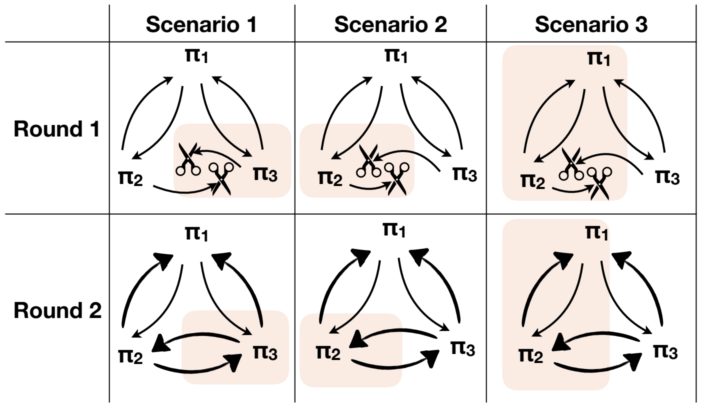

Figure 2: The scenarios from the proof. All protocols are executed as specified. A shade on the background of a protocol indicates that the corresponding party is corrupted (the adversary still executes the respective protocol on the honest input, but may drop some messages). A solid arrow indicates that the message that the corresponding protocol would send is indeed sent; cut arrows indicate that respective messages are dropped, where we illustrate which adversarial behavior is the reason for dropping a message by scissors; bold arrows indicate that this second-round message depends on the protocol having seen some incomplete transcript (due to dropped messages) in the first round and might therefore adapt its behavior accordingly.

Scenario 1: The adversary corrupts only P_3 and has him play π_3, but drops the message $m_{1,3\to 2}$ that π_3 sends to P_2 in the first round (i.e., the message is never delivered to π_2) and does not deliver to π_3 the message that it receives from P_2 in the first round. Other than this intervention, all machines execute their prescribed code and all other messages are sent and delivered as specified by the protocol π. In particular, the instance of π_3 which the adversary emulates is not aware that the message $m_{1,3\to 2}$ (which it generated and tried to send to π_2 in the first round) was never delivered, and it is not aware that P_2 did send it a message $m_{1,2\to 3}$ in the first round but it was blocked. In other words the internal state of π_2 reflects the fact that (resp., π_3) the message to π_3 (resp., π_2) is sent, but the message from π_3 (resp., π_2) did not arrive.

Scenario 2: The adversary corrupts only P_2 and has him play π_2 with the modification that he drops the first-round message $m_{1,3\to 2}$ received from P_3 (again, the message is never delivered to π_2) and the message $m_{1,2\to 3}$ that π_2 sends to P_3. Other than this specific intervention, all machines execute their prescribed code and all other messages are sent and delivered as specified by the protocol π. In particular, the simulated instance of π_2 is not aware that its first round message $m_{1,2\to 3}$ for P_3 was never delivered, and it is not aware that P_3 did send it a message $m_{1,3\to 2}$ in the first round but it was blocked, as above.

Scenario 3: The adversary corrupts P_2 and P_3. Both parties play exactly the same protocol as in Scenario 2.

First we observe the following: In all three scenarios the three machines witness the same interaction— i.e., their (joint) internal states are identically distributed. Indeed, all three adversarial strategies have the effect of execution of the prescribed protocol without the first message from π_3 to π_2 and from π_2 to π_3. Since π_1, π_2, and π_3 are protocol-machines (interactive algorithms), their behavior cannot depend on who is corrupted. This means that their (joint) output (distribution) in Scenario 1 must be indistinguishable (in fact, identically distributed) to their output in Scenarios 2 and 3.

Now consider an execution of this protocol on uniformly random inputs. We consider the following two cases for Scenario 1, where the probabilities are defined over the choice of the correlated randomness, the random coins used by the protocols, and the randomness used for selecting the inputs, and analyze them in turn.

Case 1: The honest parties abort (with noticeable probability). We prove that if an abort occurs with noticeable probability, then the security of the protocol is violated: Due to the identifiability requirement, if in Scenario 1 there is an abort, then both π_1 and π_2 need to output the identity of P_3 (as a cheater) as he is the only corrupted party. However, since as argued above the output distributions in the two scenarios are indistinguishable, the fact that in Scenario 1, π_1 aborts with the identity of P_3 with noticeable probability implies that also in Scenario 2, π_1 will also abort identifying P_3 with noticeable probability.

By the assumption that π is secure with identifiable abort—which implies that honest parties agree on the identity of a corrupted party in case of abort—the latter statement implies that in Scenario 2, with noticeable probability, π_3 will abort with the same cheater, i.e., the honest party P_3 (who is running π_3) will abort identifying itself as a cheater contradicting the fact that π is secure with identifiable abort. (Security with identifiable abort only allows an abort identifying a corrupted party.) This means that the protocol cannot abort with noticeable probability which leaves Case 2, below, as the only alternative.

Case 2: The honest parties do not abort (with overwhelming probability). We prove that an adversary corrupting P_1 in addition to P_2 can learn both $x_{3,1}$ and $x_{3,2}$ with noticeable probability, which is impossible in an ideal evaluation of $\hat{f}$, as follows. Observe that since, in this case, the probability of aborting in Scenario 1 is negligible and the joint views of the parties are indistinguishable between the two scenarios, the probability that an abort occurs in Scenario 2 or Scenario 3 is also negligible. Furthermore, because Scenario 3 consist of the same protocols in exactly the same configuration and with the same messages dropped, the output of the protocols in Scenario 3 is distributed identically to the output of the protocol in Scenario 2, namely it is the output of the function on the actual inputs of P_1 and P_3 and some input from P_2.

Next, observe that the security of π for this case implies that for every adversary in Scenario 2 there exists a simulator corrupting P_2. Let $\mathcal{A}_2$ denote the adversary that chooses an input for π_2 uniformly at random and plays the strategy specified in Scenario 2, and let $\mathcal{S}_2$ denote the corresponding simulator. Denote by X_2^* the random variable corresponding to the input x_2^* that $\mathcal{S}_2$ hands to the functionality for $\hat{f}$ on behalf of P_2, and denote by $X_1 = (X_{1,1}, X_{1,2})$ and $X_3 = (X_{3,1}, X_{3,2})$ the random variables corresponding to the inputs of the honest parties. The following claim states that X_2^* might take any of the values 0 or 1 with noticeable probability.

Claim 3.10. *For each* $b \in \{0, 1\}$, $\Pr[X_2^* = b]$ *is noticeable.*

Proof. First we note that due to input independence—i.e., because in the ideal experiment the simulator needs to hand inputs corresponding to the corrupted parties before seeing any information about the honest parties' inputs—it must hold that $\Pr[X_2^* = b] = \Pr[X_2^* = b \mid X_1, X_3]$. Hence, it suffices to prove that $\Pr[X_2^* = x_2^* \mid X_1, X_3]$ is noticeable for each of the two possible input choices $x_2^* \in \{0,1\}$ for the simulator. Assume towards a contradiction that this is not true. This means that with overwhelming probability the simulator always inputs the same $x_2^* = b$. Without loss of generality, assume that $b = 0$ (the argument for $b = 1$ is symmetric). Since the protocol aborts only with negligible probability, security implies that the distribution of the public output for every P_i with this simulator $\mathcal{S}_2$ is (computationally) indistinguishable from $f(X_1, 0, X_3) = X_{1,1}^{\kappa} \oplus X_{3,(1+X_{1,2})}$.

However, since $\mathcal{S}_2$ is a simulator for π with adversary $\mathcal{A}_2$ who uses a uniform input in his π_2 emulation, this implies that the interaction of the protocols π_1, π_2, and π_3 in Scenario 2 must also have as public output a value with distribution indistinguishable from $X_{1,1}^{\kappa} \oplus X_{3,(1+X_{1,2})}$. Now, using the fact that the views which the protocol machines in Scenario 2 and 1 are indistinguishable,[10] we can deduce that the public output in Scenario 1 needs to also be distributed indistinguishably from $X_{1,1}^{\kappa} \oplus X_{3,(1+X_{1,2})}$.

However, in Scenario 1, party P_2 is not corrupted which means that the public output distribution needs to be indistinguishable from $f(X_1, X_2, X_3^*)$, where $X_3^* = (X_{3,1}^*, X_{3,2}^*)$ is the input distribution of the simulator $\mathcal{S}_3$ for the corrupted P_3, existence of which is implied by the security of π. But this means that $\mathcal{S}_3$ will have to come up with X_3^* such that the public-output distribution $f(X_1, X_2, X_3^*) = X_{1,1}^{\kappa} \oplus X_2^{\kappa} \oplus X_{3,1+(X_{1,2} \oplus X_2)}^*$ is distributed indistinguishably from $X_{1,1}^{\kappa} \oplus X_{3,(1+X_{1,2})}^*$. Since X_3^* cannot depend on X_1 or X_2, this is impossible. □

The following claim follows directly from Claim 3.10 and the security of π (recall that we are under the assumption that Scenario 2 terminates without abort except with negligible probability).

Claim 3.11. *For any inputs x_1 and x_3 for protocols π_1 and π_3 in Scenario 2, the probability (over the input-choice of x_2 and the local randomness r_2 given to π_2) that the public output is $x_{1,1}^{\kappa} \oplus x_2^{\kappa} \oplus x_{3,1}$ (i.e., $x_{1,2} = x_2$) is noticeable, and so is the probability that the public output $x_{1,1}^{\kappa} \oplus x_2^{\kappa} \oplus x_{3,2}$ (i.e., $x_{1,2} \neq x_2$).*

The final claim that we prove provides the attack discussed at the beginning of the proof for Case 2.

Claim 3.12. *An adversary $\mathcal{A}$ corrupting both P_1 and P_2 can learn both $x_{3,1}$ and $x_{3,2}$ with noticeable probability.*

Proof. We consider the following adversary $\mathcal{A}$ corrupting parties P_1 and P_2. $\mathcal{A}$ uses the protocols π_1 and π_2 as follows:

- During the first (point-to-point) round of π: $\mathcal{A}$ chooses uniform input and randomness (x_1, r_1) for π_1, computes π_1's first-round messages $m_{1,1\to 2}$ and $m_{1,1\to 3}$ for π_2 and π_3 and sends $m_{1\to 3}$ to P_3 on behalf of P_1. $\mathcal{A}$ does not send anything to π_3 on behalf of P_2 in this first round. $\mathcal{A}$ receives the first-round messages from P_3 for both P_1 and P_2, denoted $m_{1,3\to 1}$ and $m_{1,3\to 2}$, respectively. This completes the first round.
- In the second round, $\mathcal{A}$ (playing a rushing strategy) receives π_3's broadcast message, denote it as $\bar{m}_{2,3\to *}$, and conducts the following experiment:

[10] Note that although parties P_3 and P_2 are corrupted in these scenarios, the corresponding adversary still executes π_3 and π_2, respectively and has some transmitted message dropped. Hence, we can define the view of these protocols in this concrete attack scenario although they are controlled by the adversary.

1. It simulates two independent executions of π_2 with independent uniformly chosen (input,randomness)-pairs $(x_2^{(1)}, r_2^{(1)})$ and $(x_2^{(2)}, r_2^{(2)})$. We refer to the execution with input and randomness $(x_2^{(i)}, r_2^{(i)})$ as "Execution i." Each Execution i yields a first-round message $m_{1,2\to 1}^{(i)}$ and $m_{1,2\to 3}^{(i)}$ that π_2 on input $(x_2^{(i)}, r_2^{(i)})$ would send to π_1 and π_3, respectively, in the first round of π.
2. $\mathcal{A}$ ignores $m_{1,2\to 3}^{(i)}$ and simulates sending $m_{1,2\to 1}^{(i)}$ to π_1 as π_2's first-round message to π_1. $\mathcal{A}$ also delivers $m_{1,1\to 2}$ to π_2 as π_1's first-round message in both executions.
3. In each Execution i, $\mathcal{A}$ computes the messages $\bar{m}_{2,2\to *}^{(i)}$ and $\bar{m}_{2,1\to *}^{(i)}$ that π_2 and π_1 would broadcast in the second round. It hands $\bar{m}_{2,3\to *}$ and $\bar{m}_{2,2\to *}^{(i)}$ (resp., $\bar{m}_{2,1\to *}^{(i)}$) to π_1 (resp., π_2) as the broadcast round messages of π_3 and π_2 (resp., π_1) and computes the output of π_1 and π_2.
4. $\mathcal{A}$ denotes the public output of π_1 and π_2 in Execution i as $y^{(i)}$ and their private outputs as $y_1^{(i)} = (y_{1,1}^{(i)}, y_{1,2}^{(i)})$ and $y_2^{(i)}$, respectively; $\mathcal{A}$ computes and outputs $z^{(1)} = y^{(1)} \oplus (y_{1,1}^{(1)})^{\kappa} \oplus (y_2^{(1)})^{\kappa}$ and $z^{(2)} = y^{(2)} \oplus (y_{1,1}^{(2)})^{\kappa} \oplus (y_2^{(2)})^{\kappa}$.

By design of the above experiment, one can verify that the joint view of π_1 and π_2 in Execution i is identically distributed to their joint view in Scenario 3 with inputs x_1 and x_3 for P_1 and P_3, respectively, and uniformly chosen input and randomness for π_2. This, in turns (as discussed in the beginning of the proof) is identically distributed to the joint view in Scenario 2 with inputs x_1 and x_3 for P_1 and P_3, respectively, and uniformly chosen input and randomness for π_2. Hence, $y_1^{(1)} = y_1^{(2)} = x_1$,[11] and $y^{(i)} = f(x_1, x_2^{(i)*}, x_3)$, where $x_2^{(i)*}$ is the effective input of the adversary which is distributed indistinguishably from the random variable X_2^* from Claim 3.10 (i.e., it has noticeable probability to take any of the values 0 or 1). Using Claim 3.11 and the definition of the function this implies that, with noticeable probability, $z^{(1)}$ and $z^{(2)}$ are the two private inputs $x_{3,1}$ and $x_{3,2}$ of π_3 (in some order). □

Finally, we observe that, by the definition of the function, the probability that a simulator $\mathcal{S}$ for the adversary $\mathcal{A}$ from Claim 3.12 (who corrupts P_1 and P_2) outputs both inputs of π_3 is negligible. Hence, Claim 3.12 contradicts the assumed security of π. □

4 Feasibility of Two-Round MPC with Limited Use of Broadcast

In this section, we present our feasibility results, showing how to compute any function with unanimous abort when only the second round of the MPC protocol is over broadcast, and with selective abort purely over pairwise channels. More formally:

Theorem 4.1. *Assume the existence of a two-round maliciously secure OT protocol, let f be an efficiently computable n-party function, and let $t < n$. Then,*

1. *f can be securely computed with unanimous abort, tolerating a PPT static, malicious t-adversary, by a two-round protocol in which the first round is over private channels and the second over broadcast.*

[11]Observe that the security definition would, in principle, allow that a corrupted P_1 would induce a different effective input x_1^* than his original input x_1, as long as jointly with P_2's input they induce the same distribution of f; this however cannot be the case with the above adversary for our function, as π_1 is required to include its own input to its private output and, from the indistinguishability of Scenarios 2 and 3, this private output has to be his honest input.

2. *f can be securely computed with selective abort, tolerating a PPT static, malicious t-adversary, by a two-round protocol over private channels.*

The proof of Theorem 4.1 follows from Lemmas 4.3 and 4.9 (proven in Sections 4.2 and 4.3, respectively) that show how to compile any two-broadcast-round protocol secure with unanimous (resp., selective) abort by a black-box straight-line simulation, to the desired result. Theorem 4.1 follows from that fact, and the two-broadcast-round MPC protocols presented in [32, 10].

The only cryptographic assumption used in our compiler is a garbling scheme that is used to garble the second-round next-message function of the protocol. As observed in [2], for the protocol from [32] the second-round next-message function is in NC^1. Therefore, by using information-theoretic garbling schemes for NC^1 [46, 47] and the information-theoretic two-broadcast-round protocol of [34] (in the OT-correlation model, where parties receive correlated randomness for precomputed OT [6]), we obtain the following corollary.

Corollary 4.2. *Let f be an efficiently computable n-party function and let $t < n$. Then,*

1. *f can be computed with information-theoretic security and unanimous abort in the OT-correlation model, tolerating a static, malicious t-adversary, by a two-round protocol in which the first round is over private channels and the second over broadcast.*
2. *f can be computed with information-theoretic security and selective abort in the OT-correlation model, tolerating a static, malicious t-adversary, by a two-round protocol over private channels.*

Structure of two-round protocols. Before proving Theorem 4.1, we present the notations that will be used for the proof. We consider n-party protocols defined in the correlated-randomness hybrid model, where a trusted party samples $(r_1, \ldots, r_n) \leftarrow D_{\mathsf{corr}}$ from some predefined efficiently sampleable distribution D_{corr}, and each party P_i receives r_i at the onset of the protocol. For simplicity, and without loss of generality, we assume that the random coins of each party are a part of the correlated randomness. The probabilities below are over the random coins for sampling the correlated randomness and the random coins of the adversary.

The two-round n-party protocol is then defined by the set of three functions per party $\{(\mathsf{first\text{-}msg}_i, \mathsf{second\text{-}msg}_i, \mathsf{output}_i)\}_{i \in [n]}$. Every party P_i operates as follows:

- The first-round messages are computed by the function $(m^1_{i\to 1}, \ldots, m^1_{i\to n}) = \mathsf{first\text{-}msg}_i(x_i, r_i)$, which is a deterministic function of his input x_i and randomness r_i. If the first round is over broadcast it holds that $m^1_{i\to 1} = \ldots = m^1_{i\to n}$, and we denote the unique message as m^1_i.
- The second-round messages are computed by the next-message function $(m^2_{i\to 1}, \ldots, m^2_{i\to n}) = \mathsf{second\text{-}msg}_i(x_i, r_i, m^1_{1\to i}, \ldots, m^1_{n\to i})$, which is a deterministic function of x_i, r_i and the first-round message $m^1_{j\to i}$ received from each P_j. As before, if the second round is over broadcast we denote the unique message as m^2_i.
- The output is computed by the function $y = \mathsf{output}_i(x_i, r_i, m^1_{1\to i}, \ldots, m^1_{n\to i}, m^2_{1\to i}, \ldots, m^2_{n\to i})$, which is a deterministic function of x_i, r_i and the first-round and second-round messages.

4.1 Compiling Two-Broadcast-Round Protocols

In this section, we present a compiler which transforms a two-broadcast-round MPC protocol into a two-round protocol suitable for a point-to-point network. The compiler is based on the compiler presented in Ananth et al. [2], which considered information-theoretic honest-majority protocols that are executed over both private point-to-point channels and a broadcast channel. We adapt this compiler to the dishonest-majority setting, where the input protocol is defined purely over a broadcast channel. The compiler is presented in Figure 3.

Let π_{bc} be a two-round MPC protocol in the broadcast model. Initially, every party "hard-wires" his input and randomness to the circuit computing the second-round next-message function $\mathsf{second\text{-}msg}_{i,x,r}(m_1, \ldots, m_n)$ on the first-broadcast-round messages. Next, each party garbles this circuit and secret-shares each label using an additive secret-sharing scheme.

In the first round, each party sends to each other party over private channels[12] his first-round message from π_{bc} and one share of each garbled label. Note that all of these "adjusted" second-round circuits (one circuit generated by each party) should receive the same input values, i.e., the first-broadcast-round messages. For each input wire, corresponding to one broadcast bit, each party receives two shares (one for value 0 and the other for value 1). In the second round, every party sends to all other parties the garbled circuit as well as one share from each pair, according to the messages received in the first round. Since each party sends the same second-round message to all others, each party can either send the second-round message over a broadcast channel (in which case it is guaranteed that all parties receive the same messages) or multicast the message over (authenticated) point-to-point channels.

Next, every party reconstructs all garbled labels and evaluates each garbled circuit to obtain the second-round messages of π_{bc}. Using these messages each party can recover the output value from π_{bc}.

4.2 Unanimous Abort with a Single Broadcast Round

We start by proving that the compiled protocol $\pi = \mathsf{Comp}(\pi_{\mathsf{bc}})$ of Figure 3 in (see Figure 3) is secure with unanimous abort when the second-round message is over a broadcast channel. Intuitively, if all honest parties receive the same "common part" of the first-round message (corresponding to the first broadcast round of π_{bc}), they will be able to reconstruct the garbled labels and obtain the second-round message of each party by evaluating the garbled circuits. Note that since the second round is over broadcast, it is guaranteed that all honest parties will evaluate the same garbled circuits using the same garbled inputs, and will obtain the same output value. If there exist a pair of parties that received different first-round messages, then none of the parties will be able to reconstruct the correct labels.

The security of the compiled protocol reduces to the security of the broadcast-model protocol; however, some subtleties arise in the simulation. The simulation of the garbled circuits requires the simulated second-round messages for π_{bc} (as this is the output from the garbled circuit). To simulate the second-round message of π_{bc}, the simulator must obtain the output value that corresponds to the input values that are extracted from the corrupted parties in the first round. However, since the adversary can send different first-round messages to different honest parties over the point-to-point channels, there may be multiple input values that can be extracted—in fact, the messages received by every honest party can define a different set of input values for the corrupted parties.

In more detail, given an adversary $\mathcal{A}$ for the compiled protocol π, we construct a simulator $\mathcal{S}$. We would like to use the security of π_{bc} to simulate the "common part" of the honest parties' messages. However, the adversary $\mathcal{A}$ induces *multiple adversaries* for π_{bc}, one for every honest party. For every honest party P_j we define a *receiver-specific adversary* $\mathcal{A}_j$ for π_{bc}, by forwarding the first-broadcast-round messages to $\mathcal{A}$ and responding with the messages $\mathcal{A}$ sends to P_j (recall that $\mathcal{A}$ can send different messages to different honest parties in π). By the security of π_{bc}, for every such $\mathcal{A}_j$ there exists a simulator $\mathcal{S}_j$.

[12]Private channels can be realized over authenticated channels without additional rounds assuming a public-key infrastructure (PKI) for public-key encryption.

Protocol $\pi = \mathsf{Comp}(\pi_{\mathsf{bc}})$

- **Common input:** A two-broadcast-round protocol π_{bc}, represented by the set of functions $\{\mathsf{first\text{-}msg}_i, \mathsf{second\text{-}msg}_i, \mathsf{output}_i\}_{i \in [n]}$ and a garbling scheme $(\mathsf{Garble}, \mathsf{Eval})$.
- **Private input:** Every party P_i has a private input $x_i \in \{0,1\}^*$.
- **Correlated randomness:** The correlated randomness $(r_1, \ldots, r_n) \leftarrow D^{\pi_{\mathsf{bc}}}_{\mathsf{corr}}$ (for π_{bc}) is sampled at the onset of the protocol, and every party P_i receives r_i.
- **Notation:** For every $i \in [n]$, denote by $C_{i,x,r}(m_1, \ldots, m_n)$ the Boolean circuit with hard-wired values x and r that upon receiving n inputs $m_1, \ldots, m_n$, computes $\mathsf{second\text{-}msg}_i(x, r, m_1, \ldots, m_n)$. For simplicity, assume that each first-round message is ℓ-bits long, hence each such circuit has $L = n \cdot \ell$ inputs bits.
- **The protocol:**
  - **First round:** Every party P_i proceeds as follows.
    1. Let $m_i^1 = \mathsf{first\text{-}msg}_i(x_i, r_i)$ be P_i's first-broadcast-round message in π_{bc}.
    2. Compute $(\mathsf{GC}_i, \boldsymbol{K}_i) \leftarrow \mathsf{Garble}(1^\kappa, C_{i,x,r})$, where $\boldsymbol{K}_i = (K^0_{i,1}, K^1_{i,1}, \ldots, K^0_{i,L}, K^1_{i,L})$.
    3. For every $\lambda \in [L]$ and $b \in \{0,1\}$, sample n uniformly random strings $K^b_{i \to 1, \lambda}, \ldots, K^b_{i \to n, \lambda}$, conditioned on $K^b_{i,\lambda} = \bigoplus_{j \in [n]} K^b_{i \to j, \lambda}$.
    4. Send to every party P_j the message $(m_i^1, \{K^0_{i \to j, \lambda}, K^1_{i \to j, \lambda}\}_{\lambda \in [L]})$.
  - **Second round:** In case party P_i did not receive a message from some other party, he aborts; otherwise, P_i proceeds as follows.
    1. Let $(m^1_{j \to i}, \{K^0_{j \to i, \lambda}, K^1_{j \to i, \lambda}\}_{\lambda \in [L]})$ be the first-round message received from P_j.
    2. Denote the concatenation of all the messages $m^1_{j \to i}$ as
    $$(\mu_{i,1}, \ldots, \mu_{i,L}) \coloneqq (m^1_{1 \to i}, \ldots, m^1_{n \to i}) \in \{0,1\}^L.$$
    3. Send to all the parties the message $(\mathsf{GC}_i, \{K^{\mu_{i,\lambda}}_{1 \to i, \lambda}\}_{\lambda \in [L]}, \ldots, \{K^{\mu_{i,\lambda}}_{n \to i, \lambda}\}_{\lambda \in [L]})$.
  - **Output:** In case party P_i did not receive a message from some other party, he aborts; otherwise, P_i proceeds as follows.
    1. Let $(\mathsf{GC}_j, \{K_{1 \to j, \lambda}\}_{\lambda \in [L]}, \ldots, \{K_{n \to j, \lambda}\}_{\lambda \in [L]})$ be the second-round message received from party P_j.
    2. For every $j \in [n]$ and $\lambda \in [L]$, reconstruct each garbled label as
    $$K_{j,\lambda} = \bigoplus_{h \in [n]} K_{j \to h, \lambda}.$$
    3. For every $j \in [n]$, evaluate the garbled circuit received from P_j as
    $$m_j^2 = \mathsf{Eval}(\mathsf{GC}_j, K_{j,1}, \ldots, K_{j,L}).$$
    (If any of the evaluations fails, abort.)
    4. Compute and output $y = \mathsf{output}_i(x_i, r_i, (m^1_{1 \to i}, \ldots, m^1_{n \to i}), (m^2_1, \ldots, m^2_n))$.

Figure 3: *Compiling two-broadcast-round MPC protocols into two P2P rounds.*

To define the simulator $\mathcal{S}$ (for the adversary $\mathcal{A}$), we use one of the simulators $\mathcal{S}_j$ corresponding to the honest parties (the choice of which simulator to use is arbitrary). $\mathcal{S}$ initially receives from $\mathcal{S}_j$ either the corrupted parties' inputs or an abort message, and forwards the received message to the trusted party. If $\mathcal{S}_j$ does not abort, $\mathcal{S}$ receives back the output value y, forwards y to $\mathcal{S}_j$ and receives the simulated second-round messages from $\mathcal{S}_j$'s output. Next, $\mathcal{S}$ invokes $\mathcal{A}$ and simulates the first-round messages of π (using the simulated first-round messages for π_{bc} obtained from $\mathcal{S}_j$), receives back the first-round messages from $\mathcal{A}$, and checks whether these messages are consistent. If so, $\mathcal{S}$ completes the simulation by constructing simulated garbled circuits that output the correct second-round messages (if $\mathcal{A}$'s messages are consistent, the simulated messages by $\mathcal{S}_j$ are valid for all honest parties). If $\mathcal{A}$'s messages are inconsistent, $\mathcal{S}$ simulates garbled circuit that output dummy values (e.g., zeros), which is ok since the $\mathcal{A}$ will not learn the labels to open them.

Lemma 4.3. *Let π_{bc} be a two-broadcast-round protocol secure with unanimous abort by a black-box straight-line simulation and assume that garbling schemes exist. Consider the protocol $\pi = \mathsf{Comp}(\pi_{\mathsf{bc}})$ where the first round is over secure point-to-point channels and the second round is over broadcast. Then, π is secure with unanimous abort.*

Proof. Let $\mathcal{A}$ be an adversary attacking protocol π and let $\mathcal{I} \subseteq [n]$. Without loss of generality, we assume that $\mathcal{A}$ is deterministic and that the output of $\mathcal{A}$ consists of his entire view during the protocol, i.e., the auxiliary information, the input and correlated randomness of all corrupted parties, and the messages received by honest parties during the protocol. Let SimGC be the simulator for the garbling scheme. We will construct a simulator $\mathcal{S}$ as follows.

Receiver-specific adversaries. We start by defining the class of adversarial strategies $\{\mathcal{A}_j\}_{j\notin\mathcal{I}}$ for the protocol π_{bc}. For every $j \notin \mathcal{I}$, adversary $\mathcal{A}_j$ starts by invoking $\mathcal{A}$ on his input x_i correlated randomness r_i and auxiliary information aux.

– Upon receiving the first-broadcast-round message m_h^1 from an honest party P_h in π_{bc}, $\mathcal{A}_j$ samples a uniformly random κ-bit string $K_{h\to i,\lambda}^b$ for every $i \in \mathcal{I}$, every $\lambda \in [L]$, and every $b \in \{0,1\}$. Next, $\mathcal{A}_j$ sends to $\mathcal{A}$ the message $(m_h^1, \{K_{h\to i,\lambda}^0, K_{h\to i,\lambda}^1\}_{\lambda\in[L]})$ on behalf of the honest P_h to every corrupted P_i over the point-to-point channel in π. For every $g \notin \mathcal{I}$, denote $m_{h\to g}^1 = m_h^1$.

  Let $(m_{i\to h}^1, \{K_{i\to h,\lambda}^0, K_{i\to h,\lambda}^1\}_{\lambda\in[L]})$ be the message sent by $\mathcal{A}$ on behalf of every corrupted P_i to every honest P_h in π. $\mathcal{A}_j$ broadcasts the message $m_{i\to j}^1$ for every corrupted party P_i in the protocol π_{bc} (i.e., the messages sent by corrupted parties to P_j in π).

– Upon receiving the second-round message m_h^2 from an honest party P_h in π_{bc}, $\mathcal{A}_j$ invokes the garbling-scheme simulator to obtain $(\mathsf{GC}_h, K_{h,1}, \ldots, K_{h,L}) \leftarrow \mathsf{SimGC}(1^\kappa, C_h, m_h^2)$ (where C_h is the circuit computing $\mathsf{second\text{-}msg}_h$ with input and randomness set to 0). Denote

$$(\mu_{h,1}, \ldots, \mu_{h,L}) \coloneqq (m_{1\to h}^1, \ldots, m_{n\to h}^1) \in \{0,1\}^L.$$

  Next, for every $\lambda \in [L]$ and $i \in \mathcal{I}$, denote $K_{h\to i,\lambda} = K_{h\to i,\lambda}^{\mu_{h,\lambda}}$; $\mathcal{A}_j$ samples random shares $K_{h\to g,\lambda}$ for $g \notin \mathcal{I}$ conditioned on $K_{h,\lambda} = \bigoplus_{i\in[n]} K_{h\to i,\lambda}$, and sends to $\mathcal{A}$ the message $(\mathsf{GC}_h, \{K_{1\to h,\lambda}\}_{\lambda\in[L]}, \ldots, \{K_{n\to h,\lambda}\}_{\lambda\in[L]})$ as the second-round broadcast message of P_h in π.

  Let $(\mathsf{GC}_i, \{K_{1\to i,\lambda}\}_{\lambda\in[L]}, \ldots, \{K_{n\to i,\lambda}\}_{\lambda\in[L]})$ be the broadcast message received from $\mathcal{A}$ on behalf of P_i; denote $K_{i\to h,\lambda} = K_{i\to h,\lambda}^{\mu_{h,\lambda}}$ (based on the first-round message from P_i), let $K_{i,\lambda} = \bigoplus_{h\in[n]} K_{i\to h,\lambda}$, and compute $m_i^2 = \mathsf{Eval}(\mathsf{GC}_i, K_{i,1}, \ldots, K_{i,L})$.

– Finally, $\mathcal{A}_j$ broadcasts the messages m_i^2 for every corrupted P_i in π_{bc}, outputs whatever $\mathcal{A}$ outputs, and halts.

By the security of π_{bc}, for every $j \notin \mathcal{I}$ there exists a simulator $\mathcal{S}_j$ for the adversarial strategy $\mathcal{A}_j$ such that for every auxiliary information aux and input vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ it holds that

$$\text{REAL}_{\pi_{\mathsf{bc}},\mathcal{A}_j(\mathsf{aux}),\mathcal{I}}(\kappa, \boldsymbol{x}) \stackrel{\mathrm{c}}{\equiv} \text{IDEAL}^{\mathsf{un\text{-}abort}}_{f,\mathcal{S}_j(\mathsf{aux}),\mathcal{I}}(\kappa, \boldsymbol{x}).$$

Every simulator $\mathcal{S}_j$ starts by sending input values to his trusted party $\boldsymbol{x}_j' = \{x_{i,j}'\}_{i\in\mathcal{I}}$. Upon receiving the output value y, the simulator $\mathcal{S}_j$ sends a message $\mathsf{abort/continue}$, and finally outputs the simulated view of the adversary, consisting of its input and the simulated messages of π_{bc}:

$$\text{VIEW}_j = (\hat{\mathsf{aux}}^j, \{(\hat{x}_i^j, \hat{r}_i^j)\}_{i \in \mathcal{I}}, \hat{m}_1^{1,j}, \ldots, \hat{m}_n^{1,j}, \hat{m}_1^{2,j}, \ldots, \hat{m}_n^{2,j}).$$

The simulator. Denote by $\mathcal{S}_{\mathsf{RS}} = \mathcal{S}_j$ for the minimal $j \notin \mathcal{I}$ (RS stands for *receiver specific*). The simulator $\mathcal{S}$ starts by invoking $\mathcal{S}_{\mathsf{RS}}$ on his input, and receiving back the input values $\boldsymbol{x}' = \{x_i'\}_{i \in \mathcal{I}}$ or an abort message. $\mathcal{S}$ forwards whatever he received to the trusted party. If $\mathcal{S}_{\mathsf{RS}}$ did not abort, $\mathcal{S}$ receives back the output value y and forwards y to $\mathcal{S}_{\mathsf{RS}}$. Next, $\mathcal{S}_{\mathsf{RS}}$ outputs the simulated view

$$\text{VIEW}_{\mathsf{RS}} = (\hat{\mathsf{aux}}, \{(\hat{x}_i, \hat{r}_i)\}_{i \in \mathcal{I}}, \hat{m}_1^1, \ldots, \hat{m}_n^1, \hat{m}_1^2, \ldots, \hat{m}_n^2).$$

The simulator $\mathcal{S}$ proceeds as follows:

1. $\mathcal{S}$ invokes $\mathcal{A}$ on his inputs and the simulated correlated randomness for corrupted parties.
2. $\mathcal{S}$ samples a uniformly random κ-bit string $K_{h \to i, \lambda}^b$ for every $i \in \mathcal{I}$, every $\lambda \in [L]$, and every $b \in \{0,1\}$. Next, $\mathcal{S}$ sends the message $(m_h^1, \{K_{h \to i,\lambda}^0, K_{h \to i,\lambda}^1\}_{\lambda \in [L]})$ on behalf of the honest P_h to every corrupted P_i.
3. Let $(m_{i \to h}^1, \{K_{i \to h,\lambda}^0, K_{i \to h,\lambda}^1\}_{\lambda \in [L]})$ be the message sent by $\mathcal{A}$ on behalf of every corrupted P_i to every honest P_h.
4. If it holds that for every $i \in \mathcal{I}$ there exists a value $\hat{m}_i^1$ such that $m_{i \to h}^1 = \hat{m}_i^1$ for every $h \notin \mathcal{I}$ (i.e., $\mathcal{A}$ sent consistent messages), proceed as follows:
   - For every honest party P_h, invoke the garbling-scheme simulator on the simulated message $\hat{m}_h^2$ to obtain $(\mathsf{GC}_h, K_{h,1}, \ldots, K_{h,L}) \leftarrow \mathsf{SimGC}(1^\kappa, C_h, \hat{m}_h^2)$ (where C_h is the circuit computing $\mathsf{second\text{-}msg}_h$ with input and randomness set to 0).
   - Denote
   $$(\mu_{h,1}, \ldots, \mu_{h,L}) \coloneqq (\hat{m}_1^1, \ldots, \hat{m}_n^1) \in \{0,1\}^L.$$
   For every $\lambda \in [L]$ and $i \in \mathcal{I}$, denote $K_{h \to i,\lambda} = K_{h \to i,\lambda}^{\mu_\lambda}$. Sample random shares $K_{h \to g,\lambda}$ for $g \notin \mathcal{I}$ conditioned on $K_{h,\lambda} = \bigoplus_{i \in [n]} K_{h \to i,\lambda}$.
   - Send to $\mathcal{A}$ the message $(\mathsf{GC}_h, \{K_{1 \to h,\lambda}\}_{\lambda \in [L]}, \ldots, \{K_{n \to h,\lambda}\}_{\lambda \in [L]})$ as the second-round broadcast message of P_h in π.
5. If there exists $i \in \mathcal{I}$ and $h, h' \notin \mathcal{I}$ such that $m_{i \to h}^1 \neq m_{i \to h'}^1$ (i.e., $\mathcal{A}$ sent inconsistent messages), proceed as follows:
   - Send abort to the trusted party computing f (unless already done so).
   - For every honest party P_h, invoke the garbling-scheme simulator on a dummy value to obtain $(\mathsf{GC}_h, K_{h,1}, \ldots, K_{h,L}) \leftarrow \mathsf{SimGC}(1^\kappa, C_h, 0^L)$.
   - For every $g \notin \mathcal{I}$ let $m_{h \to g}^1 = \hat{m}_h^1$ and denote
   $$(\mu_{h,1}, \ldots, \mu_{h,L}) \coloneqq (m_{1 \to h}^1, \ldots, m_{n \to h}^1) \in \{0,1\}^L.$$
   For every $\lambda \in [L]$ and $i \in \mathcal{I}$, denote $K_{h \to i,\lambda} = K_{h \to i,\lambda}^{\mu_{h,\lambda}}$. Samples random shares $K_{h \to g,\lambda}$ for $g \notin \mathcal{I}$.
   - Sends to $\mathcal{A}$ the message $(\mathsf{GC}_h, \{K_{1 \to h,\lambda}\}_{\lambda \in [L]}, \ldots, \{K_{n \to h,\lambda}\}_{\lambda \in [L]})$ as the second-round broadcast message of P_h in π.
6. Let $(\mathsf{GC}_i, \{K_{1 \to i,\lambda}\}_{\lambda \in [L]}, \ldots, \{K_{n \to i,\lambda}\}_{\lambda \in [L]})$ be the broadcast message received from $\mathcal{A}$ on behalf of P_i. If $\mathcal{A}$ sent consistent first-round messages and $\mathcal{S}_{\mathsf{RS}}$ did not abort, denote $K_{i \to h,\lambda} = K_{i \to h,\lambda}^{\mu_\lambda}$ (based on the first-round message from P_i), and check two things:

- That all garbled circuit of the corrupted parties can be evaluated, i.e., for every $i \in \mathcal{I}$, let $K_{i,\lambda} = \bigoplus_{h\in[n]} K_{i\to h,\lambda}$, and compute $m_i^2 = \mathsf{Eval}(\mathsf{GC}_i, K_{i,1}, \ldots, K_{i,L})$.
- That the adversary sent the correct shares for the honest parties' garbled labels, i.e., for every $i \in \mathcal{I}$ and $h \notin \mathcal{I}$, check that $K_{h\to i,\lambda} = K^{\mu_\lambda}_{h\to i,\lambda}$.

If any of these checks fails, $\mathcal{S}$ sends abort to the trusted party computing f; otherwise $\mathcal{S}$ sends $\mathsf{continue}$.

7. Finally, $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs, and halts.

Proving real/ideal indistinguishability. We now turn to prove that the joint output of the honest parties and of the adversary in the ideal and real executions are computationally indistinguishable. This is done by defining a sequence of hybrid games.

The game $\textsc{hyb}^1_{\pi,\mathcal{I},\mathcal{A}}$. In this game, the simulator has access to the internal state of the trusted party computing f: it can see the input values of the honest parties and choose their output values. The simulator emulates the honest parties in the protocol π towards the adversary $\mathcal{A}$ based on their input values and sets the output for each honest party according to its output in the simulation. Clearly, $\textsc{hyb}^1_{\pi,\mathcal{I},\mathcal{A}}$ and $\textsc{real}_{\pi,\mathcal{I},\mathcal{A}}$ are identically distributed.

The game $\textsc{hyb}^2_{\pi,\mathcal{I},\mathcal{A}}$. In this game, we modify $\textsc{hyb}^1_{\pi,\mathcal{I},\mathcal{A}}$ as follows. The simulator first invokes the receiver-specific simulator $\mathcal{S}_{\mathsf{RS}}$, receives back $\boldsymbol{x} = \{x'_i\}_{i\in\mathcal{I}}$ or abort and forwards the received message to the trusted party computing f. If $\mathcal{S}_{\mathsf{RS}}$ did not abort, $\mathcal{S}$ receives back the output value y and sends it to $\mathcal{S}_{\mathsf{RS}}$. Next, $\mathcal{S}$ emulates towards $\mathcal{A}$ the honest parties on their inputs (as in $\textsc{hyb}^1$). $\mathcal{S}$ checks if $\mathcal{A}$ sends inconsistent first-round messages; if so $\mathcal{S}$ sends abort to the trusted party. Otherwise, $\mathcal{S}$ checks to see that the corrupted parties' garbled circuits from the second-round messages can be evaluated with the garbled labels sent by $\mathcal{A}$, and that $\mathcal{A}$ sent the correct shares of the honest parties' garbled labels corresponding to the first-round messages. If so, $\mathcal{S}$ sends $\mathsf{continue}$; otherwise $\mathcal{S}$ sends abort.

Claim 4.4. *$\textsc{hyb}^1_{\pi,\mathcal{I},\mathcal{A}}$ and $\textsc{hyb}^2_{\pi,\mathcal{I},\mathcal{A}}$ are computationally indistinguishable.*

Proof. The claim follows by the security of π_{bc} and the correctness of the garbling scheme.

In case $\mathcal{A}$ sends inconsistent first-round messages the honest parties in π will abort, hence also in $\textsc{hyb}^1$. In $\textsc{hyb}^2$, $\mathcal{S}$ will send abort to the trusted party in that case, ensuring the honest parties will abort in $\textsc{hyb}^2$ as well.

In case $\mathcal{A}$ sends consistent first-round messages, then the execution of π will correspond to an execution of π_{bc} in the following sense. $\mathcal{A}$ gets the garbled labels for evaluating the garbled circuits and obtains the second-round messages for π_{bc}; thus, $\mathcal{A}$ gets to learn the output value. Next, $\mathcal{A}$ sends garbled circuits and garbled labels by the corrupted parties. If these garbled circuits cannot be evaluated with the garbled labels, all honest parties in π will abort. Similarly, if $\mathcal{A}$ sends incorrect shares for honest parties' garbled labels the parties in π will abort. In either case, $\mathcal{S}$ sends abort to the trusted party. If all garbled circuits evaluate properly and produce the second-round messages for the corrupted parties, the ability to distinguish between $\textsc{hyb}^1$ and $\textsc{hyb}^2$ translates to the ability to distinguish between $\mathcal{S}_{\mathsf{RS}}$ and the receiver-specific adversary for π_{bc}. □

The game $\textsc{hyb}^3_{\pi,\mathcal{I},\mathcal{A}}$. In this game, we modify $\textsc{hyb}^2_{\pi,\mathcal{I},\mathcal{A}}$ as follows. For every honest party P_h, instead of computing the shares of the garbled labels $K^b_{h\to i,\lambda}$ such that $K^b_{h,\lambda} = \bigoplus_{i\in[n]} K^b_{h\to i,\lambda}$, the simulator sends random shares from each honest party P_h to each corrupted P_i.

- If the first-round messages received from $\mathcal{A}$ are consistent (i.e., every corrupted party sent the same "common part" to all honest parties) then denote $K_{h\to i,\lambda} = K^{\mu_\lambda}_{h\to i,\lambda}$ (where the concatenated "broadcasted" string is $(\mu_1,\ldots,\mu_L)$), sample random shares $K_{h\to g,\lambda}$ for $g \notin \mathcal{I}$ conditioned on $K_{h,\lambda} = \bigoplus_{i\in[n]} K_{h\to i,\lambda}$, and send the second-round message using these shares.
- If not, denote $K_{h\to i,\lambda} = K^{\mu_{i,\lambda}}_{h\to i,\lambda}$ (where the concatenated first-round messages P_h received is $(\mu_{i,1},\ldots,\mu_{i,L})$), sample random shares $K_{h\to g,\lambda}$ for $g \notin \mathcal{I}$ (without any constraints), and send the second-round message using these shares.

Claim 4.5. *$\textsc{hyb}^2_{\pi,\mathcal{I},\mathcal{A}}$ and $\textsc{hyb}^3_{\pi,\mathcal{I},\mathcal{A}}$ are identically distributed.*

Proof. The only difference between the hybrids is when $\mathcal{A}$ sends inconsistent first-round messages. In $\textsc{hyb}^1$, for every $h \notin \mathcal{I}$ and $\lambda \in [L]$, the adversary can choose to see either a share of $K^0_{h,\lambda}$ or $K^1_{h,\lambda}$, but he will not get sufficiently many shares to reconstruct any of the garbled labels. In $\textsc{hyb}^2$, the adversary receives random κ-bit strings that are independent of the actual garbled labels. As the additive secret sharing has perfect security, the hybrids are identically distributed. $\square$

The game $\textsc{hyb}^4_{\pi,\mathcal{I},\mathcal{A}}$. In this game, we modify $\textsc{hyb}^3_{\pi,\mathcal{I},\mathcal{A}}$ as follows. Instead of generating the garbled circuits honestly, the simulator generates simulated garbled circuits. $\mathcal{S}$ computes the second-round messages from the emulated execution with $\mathcal{A}$ as in $\textsc{hyb}^3$, i.e., the message m^2_h sent by every $h \notin \mathcal{I}$. Next, $\mathcal{S}$ computes $(\mathsf{GC}_h, K_{h,1},\ldots,K_{h,L}) \leftarrow \mathsf{SimGC}(1^\kappa, C_{h,x_h,r_h}, m^2_h)$ (where C_{h,x_h,r_h} is the circuit $\mathsf{second\text{-}msg}_h$ with hard-wires input x_h and randomness r_h).

Claim 4.6. *$\textsc{hyb}^3_{\pi,\mathcal{I},\mathcal{A}}$ and $\textsc{hyb}^4_{\pi,\mathcal{I},\mathcal{A}}$ are computationally indistinguishable.*

Proof. The proof follows by a standard hybrid argument by considering $n+1$ intermediate hybrids, where in the j'th hybrid the garbed circuits of a party P_h with $h \le j$ are simulated as in $\textsc{hyb}^4$, and for $h > j$ are computed as in $\textsc{hyb}^3$. The 0'th hybrid correspond to $\textsc{hyb}^3$ and the n'th hybrid correspond to $\textsc{hyb}^4$. Each neighboring intermediate hybrids are computationally indistinguishable by the security of the garbling scheme. $\square$

The game $\textsc{hyb}^5_{\pi,\mathcal{I},\mathcal{A}}$. In this game, we modify $\textsc{hyb}^4_{\pi,\mathcal{I},\mathcal{A}}$ as follows. Let C_h be the circuit $\mathsf{second\text{-}msg}_h$ with hard-wired input and randomness set to 0.

- If the first-round messages sent by $\mathcal{A}$ are consistent, $\mathcal{S}$ simulates the garbled circuit for C_h and the message m^2_h (as computed in $\textsc{hyb}^4$) as $(\mathsf{GC}_h, K_{h,1},\ldots,K_{h,L}) \leftarrow \mathsf{SimGC}(1^\kappa, C_h, m^2_h)$.
- If the first-round messages sent by $\mathcal{A}$ are inconsistent, $\mathcal{S}$ simulates the garbled circuit for the circuit C_h and a dummy output value (e.g., 0^L) as $(\mathsf{GC}_h, K_{h,1},\ldots,K_{h,L}) \leftarrow \mathsf{SimGC}(1^\kappa, C_h, 0^L)$.

Claim 4.7. *$\textsc{hyb}^4_{\pi,\mathcal{I},\mathcal{A}}$ and $\textsc{hyb}^5_{\pi,\mathcal{I},\mathcal{A}}$ are computationally indistinguishable.*

Proof. The proof follows by a standard hybrid argument as before, by replacing the garbled circuits one by one. When the first-round messages are inconsistent, the adversary does not learn the labels for evaluating the garbled circuits, and therefore by the security of the garbling scheme, neighboring intermediate hybrids are computationally indistinguishable. $\square$

The game $\textsc{hyb}^6_{\pi,\mathcal{I},\mathcal{A}}$. In this game, we modify $\textsc{hyb}^5_{\pi,\mathcal{I},\mathcal{A}}$ as follows. Instead of computing the honest parties' messages based on their input values, $\mathcal{S}$ uses the simulated messages as generated by $\mathcal{S}_{\mathsf{RS}}$. That is, the first-round messages from an honest P_h to a corrupted P_i consist of $\hat{m}^1_h$ and random shares for the labels. The second round message by P_h (when $\mathcal{A}$'s first-round messages are consistent) is based on $(\mathsf{GC}_h, K_{h,1},\ldots,K_{h,L}) \leftarrow \mathsf{SimGC}(1^\kappa, C_h, \hat{m}^2_h)$ (rather than $\mathsf{SimGC}(1^\kappa, C_h, m^2_h)$).

Claim 4.8. *$\textsc{hyb}^5_{\pi,\mathcal{I},\mathcal{A}}$ and $\textsc{hyb}^6_{\pi,\mathcal{I},\mathcal{A}}$ are computationally indistinguishable.*

Proof. In $\textsc{hyb}^6$, the simulator uses the simulated messages by the receiver-specific simulator $\mathcal{S}_{\mathsf{RS}}$, in particular, all honest parties' messages correspond to the messages sent to party P_j (where $j \notin \mathcal{I}$ index used to define $\mathcal{S}_{\mathsf{RS}}$). In $\textsc{hyb}^5$, $\mathcal{A}$ learns the second-round messages for π_{bc} only when sending consistent first-round messages (except for negligible probability), so all honest parties receive the same messages as P_j. The claim now follows by the security of π_{bc}, since any distinguisher between $\textsc{hyb}^5$ and $\textsc{hyb}^6$ can be used to distinguish between the simulation of $\mathcal{S}_{\mathsf{RS}}$ and the real execution with the receiver-specific adversary.

In fact, the proof relies on the ability of $\mathcal{S}_{\mathsf{RS}}$ to extract the corrupted parties' inputs from the first-round messages, and so even by "rewinding" $\mathcal{A}$, i.e., by sending to $\mathcal{A}$ the simulated first-round messages that are obtained from the output of $\mathcal{S}_{\mathsf{RS}}$, it is guaranteed that $\mathcal{A}$ will not switch the corrupted parties' inputs to different values. The ability of $\mathcal{S}_{\mathsf{RS}}$ to extract the corrupted parties' inputs from their first-round messages follows from the assumption that $\mathcal{S}_{\mathsf{RS}}$ is straight-line and black-box. Indeed, for two-round MPC protocols a straight-line and black-box simulation implies that property, since otherwise, if the corrupted parties' input can only be extracted from the second-round messages, the adversary will be able to choose which input values to use (i.e., which second-round messages to send) as a function of the output; therefore, the *input-independence* property will not be satisfied. □

This concludes the proof of Lemma 4.3 since in $\textsc{hyb}^6_{\pi,\mathcal{I},\mathcal{A}}$ the simulator does not need to have access to the internals of the trusted party computing f, and it behaves exactly as the simulator $\mathcal{S}$ for the adversary $\mathcal{A}$; hence, $\textsc{hyb}^6_{\pi,\mathcal{I},\mathcal{A}}$ and $\textsc{ideal}^{\mathsf{un\text{-}abort}}_{f,\mathcal{I},\mathcal{S}}$ are identically distributed. □

4.3 Selective Abort with Two Point-To-Point Rounds

We proceed by proving our second result, that the compiled protocol $\pi = \mathsf{Comp}(\pi_{\mathsf{bc}})$ of Figure 3 is secure with *selective* abort when the second-round message is over a point-to-point channel. The main difference from the previous case (Section 4.2) is that the adversary can send different garbled circuits to different honest parties in the second round, potentially causing them to obtain different output values, which would violate correctness (recall that the definition of security with selective abort permits some honest parties to abort while other obtain the correct output, but it is forbidden for two honest parties to obtain two different output values.)

Lemma 4.9. *Let π_{bc} be a two-broadcast-round protocol secure with unanimous abort by a black-box straight-line simulation and assume that garbling schemes exist. Consider the protocol $\pi = \mathsf{Comp}(\pi_{\mathsf{bc}})$ where both rounds are over secure point-to-point channels. Then, π is secure with selective abort.*

Proof. The proof follows in a similar way to the proof of Lemma 4.3. We will indicate the differences in the simulation and the indistinguishability proof. Let $\mathcal{A}$ be an adversary attacking protocol π and let $\mathcal{I} \subseteq [n]$.

Receiver-specific adversaries. This class of adversaries in defined as before, with the exception that the second-round messages sent by $\mathcal{A}$ from a corrupted P_i may not be the same for all honest parties. For $j \notin \mathcal{I}$, the adversary $\mathcal{A}_j$ for the protocol π_{bc}, receives from $\mathcal{A}$ all second-round messages for π, and for very $i \in \mathcal{I}$, broadcasts the messages sent by P_i to P_j. As before, by the security of π_{bc}, for every $j \notin \mathcal{I}$ there exists a simulator $\mathcal{S}_j$ for the adversarial strategy $\mathcal{A}_j$ such that

$$\textsc{real}_{\pi_{\mathsf{bc}},\mathcal{A}_j(\mathsf{aux}),\mathcal{I}}(\kappa,\boldsymbol{x}) \overset{\mathrm{c}}{\equiv} \textsc{ideal}^{\mathsf{un\text{-}abort}}_{f,\mathcal{S}_j(\mathsf{aux}),\mathcal{I}}(\kappa,\boldsymbol{x}).$$

The simulator. Denote by $\mathcal{S}_{\mathsf{RS}} = \mathcal{S}_j$ for the minimal $j \notin \mathcal{I}$. The simulator $\mathcal{S}$ starts by invoking $\mathcal{S}_{\mathsf{RS}}$ on his input. If $\mathcal{S}_{\mathsf{RS}}$ responds with abort, then $\mathcal{S}$ sets $\mathcal{J} = [n] \setminus \mathcal{I}$ and sends $(\mathsf{abort}, \mathcal{J})$ to the trusted party (meaning that no honest party will obtain the output); if $\mathcal{S}_{\mathsf{RS}}$ responds with the input values $\boldsymbol{x}' = \{x'_i\}_{i\in\mathcal{I}}$ for the corrupted parties, $\mathcal{S}$ forwards these values to the trusted party, receives back the output value y, and sends y to $\mathcal{S}_{\mathsf{RS}}$. Next, $\mathcal{S}_{\mathsf{RS}}$ outputs the simulated view

$$\textsc{view}_{\mathsf{RS}} = (\hat{\mathsf{aux}}, \{(\hat{x}_i, \hat{r}_i)\}_{i\in\mathcal{I}}, \hat{m}^1_1, \ldots, \hat{m}^1_n, \hat{m}^2_1, \ldots, \hat{m}^2_n).$$

The simulator proceeds as in the proof of Lemma 4.3 with the following differences:

1. In Step 5 (when $\mathcal{A}$ send inconsistent first-round message), instead of simply sending abort to the trusted party, $\mathcal{S}$ sets $\mathcal{J} = [n] \setminus \mathcal{I}$ and sends $(\mathsf{abort}, \mathcal{J})$.
2. In Step 6, $\mathcal{S}$ performs the check of the second-round messages sent by $\mathcal{A}$ for every honest party. Let $\mathcal{J}$ be the set of honest parties for which the check fails; $\mathcal{S}$ sends $(\mathsf{abort}, \mathcal{J})$ to the trusted party (i.e., the parties in $\mathcal{J}$ will abort, and parties in $[n] \setminus (\mathcal{I} \cup \mathcal{J})$ will receive the output).

Proving real/ideal indistinguishability. The difference in the hybrid argument used in the proof of Lemma 4.3 lies in the game $\textsc{hyb}^2_{\pi,\mathcal{I},\mathcal{A}}$. In this hybrid, the simulator changes the way the honest parties' output values are computed: from the output as obtained from the execution of π (which is the case in $\textsc{hyb}^1$) to the output of the trusted party based on the messages sent by $\mathcal{S}$.

The game $\textsc{hyb}^2_{\pi,\mathcal{I},\mathcal{A}}$ is therefore adjusted as follows. The simulator first invokes the receiver-specific simulator $\mathcal{S}_{\mathsf{RS}}$. If $\mathcal{S}_{\mathsf{RS}}$ sends abort, $\mathcal{S}$ sends $(\mathsf{abort}, [n] \setminus \mathcal{I})$ to the trusted party. If $\mathcal{S}_{\mathsf{RS}}$ sends $\boldsymbol{x} = \{x'_i\}_{i\in\mathcal{I}}$ the simulator forwards the received message to the trusted party, receives back the output value y, and sends it to $\mathcal{S}_{\mathsf{RS}}$. Next, $\mathcal{S}$ emulates towards $\mathcal{A}$ the honest parties on their inputs (as in $\textsc{hyb}^1$). $\mathcal{S}$ checks if $\mathcal{A}$ sends inconsistent first-round messages; if so $\mathcal{S}$ sends $(\mathsf{abort}, [n] \setminus \mathcal{I})$ to the trusted party. Otherwise, $\mathcal{S}$ checks *for every honest party* to see that the corrupted parties' garbled circuits from the second-round messages can be evaluated with the garbled labels sent by $\mathcal{A}$, and that $\mathcal{A}$ sent the correct shares of the honest parties' garbled labels corresponding to the first-round messages. Let $\mathcal{J}$ denote the set of honest parties for which the check failed. $\mathcal{S}$ sends $(\mathsf{abort}, \mathcal{J})$ to the trusted party.

Proving computational indistinguishability of $\textsc{hyb}^1_{\pi,\mathcal{I},\mathcal{A}}$ (as defined in the proof of Lemma 4.3) and the adjusted $\textsc{hyb}^2_{\pi,\mathcal{I},\mathcal{A}}$ follows as before with one additional subtlety. A corrupted P_i can send two different garbled circuits GC_i and $\widetilde{\mathsf{GC}}_i$ that on the same garbled labels will produce two different second-round messages m^2_i and $\tilde{m}^2_i$. The proof follows in that case since these two messages can lead to different valid (i.e., non-aborting) output values only with negligible probability. Otherwise, the adversary will be able to choose which second-round message to send for P_i in the broadcast-model protocol π_{bc}, which will lead to a different output value, *after* receiving all of the protocol's messages; in this situation, the *input-independence* property will not be satisfied. □

Acknowledgements. We would like to thank Prabhanjan Ananth, Arpita Patra, and Divya Ravi for useful discussions and comments.

Bibliography

[1] P. Ananth, A. R. Choudhuri, A. Goel, and A. Jain. Round-optimal secure multiparty computation with honest majority. In *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 395–424. Springer, Heidelberg, 2018.

[2] P. Ananth, A. R. Choudhuri, A. Goel, and A. Jain. Two round information-theoretic MPC with malicious security. In *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 532–561. Springer, Heidelberg, 2019.

[3] B. Applebaum, Z. Brakerski, and R. Tsabary. Perfect secure computation in two rounds. In *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 152–174. Springer, Heidelberg, 2018.

[4] B. Applebaum, Z. Brakerski, and R. Tsabary. Degree 2 is complete for the round-complexity of malicious MPC. In *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 504–531. Springer, Heidelberg, 2019.

[5] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Heidelberg, 2012.

[6] D. Beaver. Precomputing oblivious transfer. In *CRYPTO'95*, volume 963 of *LNCS*, pages 97–109. Springer, Heidelberg, 1995.

[7] M. Bellare, V. T. Hoang, and P. Rogaway. Foundations of garbled circuits. In *ACM CCS 2012*, pages 784–796. ACM Press, 2012.

[8] M. Ben-Or and R. El-Yaniv. Resilient-optimal interactive consistency in constant time. *Distributed Computing*, 16(4):249–262, 2003.

[9] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, 1988.

[10] F. Benhamouda and H. Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 500–532. Springer, Heidelberg, 2018.

[11] F. Benhamouda, H. Lin, A. Polychroniadou, and M. Venkitasubramaniam. Two-round adaptively secure multiparty computation from standard assumptions. In *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 175–205. Springer, Heidelberg, 2018.

[12] E. Boyle, N. Gilboa, and Y. Ishai. Group-based secure computation: Optimizing rounds, communication, and computation. In *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 163–193. Springer, Heidelberg, 2017.

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

[14] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, 2001.

[15] R. Canetti and R. Ostrovsky. Secure computation with honest-looking parties: What if nobody is truly honest? (extended abstract). In *31st ACM STOC*, pages 255–264. ACM Press, 1999.

[16] R. Canetti, O. Poburinnaya, and M. Venkitasubramaniam. Better two-round adaptive multi-party computation. In *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 396–427. Springer, Heidelberg, 2017.

[17] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th ACM STOC*, pages 11–19. ACM Press, 1988.

[18] B. Chor, M. Merritt, and D. B. Shmoys. Simple constant-time consensus protocols in realistic failure models. *Journal of the ACM*, 36(3):591–614, 1989.

[19] R. Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *18th ACM STOC*, pages 364–369. ACM Press, 1986.

[20] R. Cohen and Y. Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. *Journal of Cryptology*, 30(4):1157–1186, Oct. 2017.

[21] R. Cohen, S. Coretti, J. A. Garay, and V. Zikas. Round-preserving parallel composition of probabilistic-termination cryptographic protocols. In *ICALP 2017*, volume 80 of *LIPIcs*, pages 37:1–37:15. Schloss Dagstuhl, 2017.

[22] R. Cohen, I. Haitner, E. Omri, and L. Rotem. Characterization of secure multiparty computation without broadcast. *Journal of Cryptology*, 31(2):587–609, Apr. 2018.

[23] R. Cohen, S. Coretti, J. A. Garay, and V. Zikas. Probabilistic termination and composability of cryptographic protocols. *Journal of Cryptology*, 32(3):690–741, July 2019.

[24] R. Cohen, I. Haitner, N. Makriyannis, M. Orland, and A. Samorodnitsky. On the round complexity of randomized Byzantine agreement. In *DISC*, pages 12:1–12:17, 2019.

[25] R. Cohen, a. shelat, and D. Wichs. Adaptively secure MPC with sublinear communication complexity. In *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 30–60. Springer, Heidelberg, 2019.

[26] D. Dolev and H. R. Strong. Authenticated algorithms for Byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

[27] M. J. Fischer and N. A. Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 14(4):183–186, 1982.

[28] M. Fitzi, D. Gottesman, M. Hirt, T. Holenstein, and A. Smith. Detectable Byzantine agreement secure against faulty majorities. In *21st ACM PODC*, pages 118–126. ACM, 2002.

[29] J. A. Garay, J. Katz, C.-Y. Koo, and R. Ostrovsky. Round complexity of authenticated broadcast with a dishonest majority. In *48th FOCS*, pages 658–668. IEEE Computer Society Press, 2007.

[30] S. Garg and A. Polychroniadou. Two-round adaptively secure MPC from indistinguishability obfuscation. In *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 614–637. Springer, Heidelberg, 2015.

[31] S. Garg and A. Srinivasan. Garbled protocols and two-round MPC from bilinear maps. In *58th FOCS*, pages 588–599. IEEE Computer Society Press, 2017.

[32] S. Garg and A. Srinivasan. Two-round multiparty secure computation from minimal assumptions. In *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 468–499. Springer, Heidelberg, 2018.

[33] S. Garg, C. Gentry, S. Halevi, and M. Raykova. Two-round secure MPC from indistinguishability obfuscation. In *TCC 2014*, volume 8349 of *LNCS*, pages 74–94. Springer, Heidelberg, 2014.

[34] S. Garg, Y. Ishai, and A. Srinivasan. Two-round MPC: Information-theoretic and black-box. In *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 123–151. Springer, Heidelberg, 2018.

[35] S. Garg, P. Miao, and A. Srinivasan. Two-round multiparty secure computation minimizing public key operations. In *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 273–301. Springer, Heidelberg, 2018.

[36] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. The round complexity of verifiable secret sharing and secure multicast. In *33rd ACM STOC*, pages 580–589. ACM Press, 2001.

[37] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. On 2-round secure multiparty computation. In *CRYPTO 2002*, volume 2442 of *LNCS*, pages 178–193. Springer, Heidelberg, 2002.

[38] O. Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004. ISBN ISBN 0-521-83084-2 (hardback).

[39] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *19th ACM STOC*, pages 218–229. ACM Press, 1987.

[40] S. Goldwasser and Y. Lindell. Secure multi-party computation without agreement. *Journal of Cryptology*, 18(3):247–287, July 2005.

[41] S. D. Gordon, F.-H. Liu, and E. Shi. Constant-round MPC with fairness and guarantee of output delivery. In *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 63–82. Springer, Heidelberg, 2015.

[42] S. Halevi, Y. Lindell, and B. Pinkas. Secure computation on the web: Computing without simultaneous interaction. In *CRYPTO 2011*, volume 6841 of *LNCS*, pages 132–150. Springer, Heidelberg, 2011.

[43] C. Hazay, P. Scholl, and E. Soria-Vazquez. Low cost constant round MPC combining BMR and oblivious transfer. In *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 598–628. Springer, Heidelberg, 2017.

[44] C. Hazay, E. Orsini, P. Scholl, and E. Soria-Vazquez. Concretely efficient large-scale MPC with active security (or, TinyKeys for TinyOT). In *ASIACRYPT 2018, Part III*, volume 11274 of *LNCS*, pages 86–117. Springer, Heidelberg, 2018.

[45] M. Hirt and U. M. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology*, 13(1):31–60, Jan. 2000. doi: 10.1007/s001459910003.

[46] Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st FOCS*, pages 294–304. IEEE Computer Society Press, 2000.

[47] Y. Ishai and E. Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *ICALP 2002*, volume 2380 of *LNCS*, pages 244–256. Springer, Heidelberg, 2002.

[48] Y. Ishai, E. Kushilevitz, and A. Paskin. Secure multiparty computation with minimal interaction. In *CRYPTO 2010*, volume 6223 of *LNCS*, pages 577–594. Springer, Heidelberg, 2010.

[49] Y. Ishai, R. Ostrovsky, and V. Zikas. Secure multi-party computation with identifiable abort. In *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 369–386. Springer, Heidelberg, 2014.

[50] Y. Ishai, R. Kumaresan, E. Kushilevitz, and A. Paskin-Cherniavsky. Secure computation with minimal interaction, revisited. In *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 359–378. Springer, Heidelberg, 2015.

[51] A. R. Karlin and A. C. Yao. Probabilistic lower bounds for Byzantine agreement and clock synchronization. Unpublished manuscript, 1986.

[52] J. Katz and C.-Y. Koo. On expected constant-round protocols for Byzantine agreement. In *CRYPTO 2006*, volume 4117 of *LNCS*, pages 445–462. Springer, Heidelberg, 2006.

[53] Y. Lindell and A. Nof. A framework for constructing fast MPC over arithmetic circuits with malicious adversaries and an honest-majority. In *ACM CCS 2017*, pages 259–276. ACM Press, 2017.

[54] Y. Lindell, A. Lysyanskaya, and T. Rabin. Sequential composition of protocols without simultaneous termination. In *21st ACM PODC*, pages 203–212. ACM, 2002.

[55] Y. Lindell, B. Pinkas, N. P. Smart, and A. Yanai. Efficient constant round multi-party computation combining BMR and SPDZ. In *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 319–338. Springer, Heidelberg, 2015.

[56] Y. Lindell, N. P. Smart, and E. Soria-Vazquez. More efficient constant-round multi-party computation from BMR and SHE. In *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 554–581. Springer, Heidelberg, 2016.

[57] S. Micali. Very simple and efficient byzantine agreement. In C. H. Papadimitriou, editor, *ITCS 2017*, volume 4266, pages 6:1–6:1, 67, Jan. 2017. LIPIcs. doi: 10.4230/LIPIcs.ITCS.2017.6.

[58] P. Mukherjee and D. Wichs. Two round multiparty computation via multi-key FHE. In *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 735–763. Springer, Heidelberg, 2016.

[59] A. Patra and D. Ravi. On the exact round complexity of secure three-party computation. In *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 425–458. Springer, Heidelberg, 2018.

[60] A. Patra and D. Ravi. On the exact round complexity of secure three-party computation. Cryptology ePrint Archive, Report 2018/481, 2018.

[61] A. Patra and D. Ravi. Beyond honest majority: The round complexity of fair and robust multi-party computation. ASIACRYPT'19 (to appear), 2019.

[62] B. Pfitzmann and M. Waidner. Unconditional Byzantine agreement for any number of faulty processors. In *STACS*, volume 577 of *LNCS*, pages 339–350. Springer, 1992.

[63] W. Quach, H. Wee, and D. Wichs. Laconic function evaluation and applications. In *59th FOCS*, pages 859–870. IEEE Computer Society Press, 2018.

[64] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *21st ACM STOC*, pages 73–85. ACM Press, 1989.

[65] X. Wang, S. Ranellucci, and J. Katz. Global-scale secure multiparty computation. In *ACM CCS 2017*, pages 39–56. ACM Press, 2017.

[66] A. C.-C. Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, 1982.

A Supplementary Proofs

Corollary 3.1 ([42]). *The function f cannot be computed with selective abort by a single-round protocol tolerating one semi-honest corrupted party.*

Proof. HLP considered the model of "MPC on the Web" in which a server is permanently online, and parties appear in turns, interact with the server to provide their inputs to the computation, and disappear. In that model, they proved that there are functions that cannot be securely computed even in the semi-honest setting. This fact proves this corollary as follows: First, observe that this impossibility can be applied to f. Towards a contradiction, assume that there exists a single-round semi-honest protocol π for f. Then, a rushing adversary corrupting P_1 can wait until he receives the messages from P_2 and P_3 and, before sending his own message, compute the outcome of f on $x_1 = 0$ and on $x_1 = 1$. Clearly, one of the two evaluations will yield $x_{3,1} \oplus x_2^\kappa$ and the other $x_{3,2} \oplus x_2^\kappa$. However, in an ideal evaluation of f the simulator can produce at most one of the two values, yielding a contradiction to the security of π. □

Corollary 3.2. *For $n \geq 3$, there exist an n-party function f_n for which there is no single-round protocol π which securely computes f_n with selective abort against even a single corruption. The statement is true even if π uses a broadcast channel in its single round.*

Proof. First we observe that, as discussed above, the statement holds for f for the case of three parties by considering the honest-looking adversary that uses the strategy from Corollary 3.1. The proof then follows trivially by a player-simulation argument: Let f_n be the n-party function among parties in $\mathcal{P} = \{P_1, \ldots, P_n\}$ in which for $i \in \{1, 2, 3\}$, party P_i has the same inputs and outputs as in f, whereas any $P \in \mathcal{P} \setminus \{P_1, P_2, P_3\}$ has input and output $\perp$. It is straightforward to verify that the honest-looking adversary who corrupts P_1 and plays the same strategy as the adversary attacking the three-party evaluation of f learns both inputs of P_3 and, therefore, cannot be simulated. $\square$