Broadcast-Optimal Four-Round MPC in the Plain Model

Michele Ciampi* [3], Ivan Damgård† [1], Divya Ravi‡ [1], Luisa Siniscalchi[2], Yu Xia[3], and Sophia Yakoubov[1]

[1]Aarhus University, Denmark; {ivan, divya, sophia.yakoubov}@cs.au.dk
[2] Technical University of Denmark, Denmark; luisi@dtu.dk
[3]The University of Edinburgh, UK; {michele.ciampi, yu.xia}@ed.ac.uk

Abstract

Motivated by the fact that broadcast is an expensive, but useful, resource for the realization of multi-party computation protocols (MPC), Cohen, Garay, and Zikas (Eurocrypt 2020), and subsequently Damgård, Magri, Ravi, Siniscalchi and Yakoubov (Crypto 2021), and, Damgård, Ravi, Siniscalchi and Yakoubov (Eurocrypt 2023), focused on *so-called broadcast optimal MPC*. In particular, the authors focus on two-round MPC protocols (in the CRS model), and give tight characterizations of which security guarantees are achievable if broadcast is available in the first round, the second round, both rounds, or not at all.

This work considers the natural question of characterizing broadcast optimal MPC in the plain model where no set-up is assumed. We focus on four-round protocols, since four is known to be the minimal number of rounds required to securely realize any functionality with black-box simulation. We give a complete characterization of which security guarantees, (namely selective abort, selective identifiable abort, unanimous abort and identifiable abort) are feasible or not, depending on the exact selection of rounds in which broadcast is available.

*Supported by the Sunday Group.
†Supported by the Villum foundation.
‡Funded by the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC).

Contents

1 Introduction

Secure Multi-party Computation (MPC) [CDv88, GMW87, Yao86] allows a set of mutually distrusting parties to compute a joint function on their private inputs, with the guarantee that no adversary corrupting a subset of parties can learn more information than the output of the joint computation. The study of round complexity of MPC protocols in various settings constitutes a phenomenal body of work in the MPC literature [ACGJ18, BGJ$^+$18, BL18, CCG$^+$20, GS18, HHPV18, KO04, MW16]. However, most of the known round-optimal protocols crucially rely on the availability of a *broadcast channel*. Informally, a broadcast channel guarantees that when a message is sent, this reaches all the parties, without ambiguity.

In practice, a broadcast channel can be realized using $t+1$ rounds of point-to-point communication, where t denotes the corruption threshold (maximal number of parties the adversary can corrupt). In fact, $t+1$ rounds are necessary for any deterministic protocol that realizes broadcast [DS83, FL82]. An alternate way of realizing broadcast would be by means of a physical or external infrastructure, e.g., a public ledger such as blockchain. Both these approaches of realizing broadcast are quite demanding and expensive; therefore it is important to minimize its use.

Driven by this motivation, a very recent line of works [CGZ20, DMR$^+$21, DRSY23] studies if it is plausible to minimize the use of broadcast while maintaining an *optimal* round complexity, at the cost of possibly settling for a weaker security guarantee. More specifically, these works investigate the best achievable guarantees when some or all of the broadcast rounds are replaced with rounds that use only point-to-point communication. All the above works focused on two-round MPC protocols where some form of setup assumption (such as a common reference string (CRS) or public-key infrastructure (PKI)) is required.

We make a study analogous to these works but in the *plain model*, where no prior setup is assumed [1]. Further, we focus on the *dishonest majority* setting where the adversary can corrupt all but one party. In this setting, *four rounds* of communication is known to be necessary [GMPP16] and sufficient [ACJ17, BGJ$^+$18, BHP17, CCG$^+$20, COWZ22, CRSW22, HHPV18] for secure computation with black-box security[2]. Notably, all the round-optimal (four-round) protocols in this setting use *broadcast* in every round. This leads us to the following natural question:

What is the trade-off between security and the use of broadcast for 4-round MPC protocols in the plain model in the dishonest majority setting?

As a first step, let us recall what kinds of security guarantees are achievable in the dishonest majority setting. The classic impossibility result of [Cle86] showed that it is in general impossible to achieve the notions of fairness (where either all or none of the parties receive the output) and guaranteed output delivery (where all the parties receive the output of the computation no matter what). In light of this, the protocols in the dishonest majority setting allow the adversary to abort prematurely and still, receive the output (while the honest parties do not). Below are the various relevant flavors of *abort security* studied in the MPC literature.

Selective Abort (SA): A secure computation protocol achieves *selective abort* if every honest party either obtains the correct output or aborts.

Selective Identifiable Abort (SIA): a secure computation protocol achieves *selective identifiable abort* if every honest party either obtains the correct output or aborts, iden-

[1] The only assumption is that authenticated communication channels are available between the parties (either broadcast or peer-to-peer channels).

[2] By black-box security we mean that the simulator has only black-box access to the adversary. As in the mentioned prior works, all our results are concerning black-box security.

tifying one corrupt party (where the corrupt party identified by different honest parties may potentially be different).

Unanimous Abort (UA): A secure computation protocol achieves *unanimous abort* if either *all* honest parties obtain the correct output, or they all (unanimously) abort.

Identifiable Abort (IA): A secure computation protocol achieves *identifiable abort* if either all honest parties obtain the correct output, or they all (unanimously) abort, *identifying one corrupt party.*

Of these notions, SA is the weakest, IA the strongest, while SIA (recently introduced in [DRSY23]) and UA are "in between", and incomparable.

1.1 Our Contributions

We settle the above question by giving a complete characterization of which of the above four security guarantees is feasible or not w.r.t. all the possible broadcast communication patterns that one can have in 4-rounds, namely, if no broadcast is available, if broadcast is available in just one (two or three) rounds, and in which one(s).

We give a concise overview of our results below, which are described in more detail in Section 1.2. We recall that our impossibility results hold w.r.t. black-box simulation, which is also the case for [GMPP16].

No Broadcast: We show that if broadcast is not used in any of the four rounds, then *selective abort* is the best notion that can be achieved.

Broadcast in One Round: We show that if broadcast is used in exactly one round, then *unanimous abort* can be achieved if it is used in the *last* round; otherwise *selective abort* continues to remain the best achievable guarantee.

Broadcast in Two Rounds: We show that if broadcast is used in exactly two rounds, the feasibility landscape remains the same as the above.

Broadcast in Three Rounds: We show that if broadcast is used in exactly three rounds, then *selective identifiable abort* can be achieved if it is used in the first three rounds; otherwise it continues to remain impossible. The feasibility of other notions does not change in this setting.

Broadcast in Four Rounds: If broadcast is used in all four rounds, the strongest notion of *identifiable abort* becomes possible [CRSW22].

In Table 1 we summarize our findings.

1.2 Technical Overview

We start by presenting the technical overview of our positive results, and in the next section, we will provide a high-level idea about how our impossibility proof works.

Broadcast Pattern	Possible?	Theorem reference
Selective Abort (SA)		
$P2P^4$	✓	Theorem 1
Identifiable Abort (IA)		
BC^4	✓	[CRSW22]
BC^3-$P2P$	✗	Theorem 9
Any other 4-round pattern	✗	Follows from the set on impossibilities for SIA, see Table 2 for the corresponding theorems.

Broadcast Pattern	Possible?	Theorem reference
Unanimous Abort (UA)		
BC^3-$P2P$	✗	Theorem 9
$P2P^3$-BC	✓	Theorem 7
Selective Identifiable Abort (SIA)		
BC^3-$P2P$	✓	Theorem 8
Any other 4-round pattern	✗	See Table 2 for the corresponding theorems.

Table 1: Complete characterization of feasibility and impossibility for 4-round dishonest majority MPC with different communication patterns in the plain model. We denote the acronym $P2P$ (resp. BC) to indicate the peer-to-peer (resp. broadcast) channel. We use the notation $P2P^x$ (resp. BC^x) to indicate x consecutive rounds of peer-to-peer (resp. broadcast) communications.

Broadcast Pattern	Implied Patterns
BC^2-$P2P$-BC ✗(Theorem 16)	BC-$P2P^2$-BC, BC-$P2P^3$, $P2P$-BC-$P2P$-BC, $P2P$-BC-$P2P^2$
BC^2-$P2P^2$ ✗(Theorem 12)	
BC-$P2P$-BC^2 ✗(Theorem 18)	BC-$P2P$-BC-$P2P$
$P2P$-BC^3 ✗(Theorem 17)	$P2P^2$-BC^2, $P2P^2$-BC-$P2P$, $P2P$-BC^2-$P2P$, $P2P^3$-BC, $P2P^4$

Table 2: Impossibility results for 4-round MPC with SIA security against dishonest majority in the plain model. The third column "Implied Patterns" means that the patterns in this column are implied by the pattern in the first column "Broadcast Patterns". An impossibility in a stronger broadcast pattern setting implies the impossibility in a weaker broadcast pattern setting, where a broadcast pattern BP1 is weaker than a pattern BP2 if BP1 replaces at least one of the broadcast rounds in BP2 with a $P2P$ round (without introducing any additional BC rounds over BP2).

1.2.1 Feasibility Results

P2P^4 SA protocol. In our first upper bound, we show that security with selective abort can be achieved when all the rounds are over $P2P$ channels. In particular, we show how to turn any protocol that is proven secure assuming that all the messages are sent over a broadcast channel, into a protocol that is secure even if all the broadcast rounds are replaced with $P2P$ rounds. As a starting point, note that if a round where a secure protocol uses broadcast (say round r) is simply replaced with peer-to-peer channels, the main problem is that the adversary can send different messages (over peer-to-peer channels) to a pair of honest parties in round r and obtain the honest parties' responses in round $(r+1)$, computed with respect to different round r messages. This potentially violates security as such a scenario would never happen in the original protocol with broadcast in round r (as the honest parties would have a consistent view of the messages sent in round r).

To ensure that honest parties' responses are obtained only if they have a consistent view of the corrupt parties' messages, the two-round construction of Cohen *et al.* [CGZ20] adopts the following trick: In addition to sending the round r message [3] over a peer-to-peer channel (as described above), the parties send a garbled circuit which computes their next-round message (by taking as input round r messages, and using the hard-coded values of input and randomness of this party) and additively share labels of this garbled circuit. In the subsequent round, parties send the relevant shares based on the round r messages they received. The main idea is that the labels corresponding to honest parties' garbled circuits can be reconstructed to obtain their round $(r+1)$ messages *only if* the adversary sends the same round r message to every honest

[3]The round r corresponds to the first round in the construction of [CGZ20].

party.

While [CGZ20] use the above idea to transform a BC-BC protocol into a $P2P$-$P2P$ protocol, we extend it to transform a BC^4 protocol to $P2P^4$ protocol. Applying the above trick of sending the next-message garbled circuits and additive shares in Round 1 and 3 will ensure that if honest parties manage to evaluate the garbled circuits in Round 2 and 4 respectively, it must be the case that the honest parties have a consistent view of the Round 1 and Round 3 messages of corrupt parties. However, there is a slight caveat: The corrupt party could still send different garbled circuits to different honest parties, say in Round 1. This will make the view of honest parties inconsistent with respect to Round 2 of the corrupt party. Note that this was not a concern in [CGZ20] as Round 2 corresponds to the last round of the protocol, unlike our case [4].

To address this, we use 'broadcast with abort' [GL05] to realize a 'weak' broadcast of garbled circuits over two peer-to-peer rounds – In the first round, as before, each party sends its garbled circuit to others. In the second round, parties additionally echo the garbled circuits they received in Round 1. A party 'accepts' a garbled circuit only if it has been echoed by all other parties, or else she aborts. This ensures that if a pair of honest parties does not abort, they must have received the same garbled circuit and therefore would have a consistent view of Round 2 of corrupt parties as well. This approach has still one issue, as it allows the adversary to send different fourth-round messages to different honest parties. We can argue that this is not a problem if the input protocol of our compiler admits a simulator that can extract the inputs of the corrupted parties in the first three rounds. This helps because if the inputs of the corrupted parties are fixed in the third round, so is the output. Intuitively, this means that no matter what fourth round the adversary sends, an honest party receiving this fourth round will either abort or compute the correct output (and all the parties will get an output generated according to the same corrupted and honest parties' inputs). Finally, we note that the protocols proposed in [BGJ$^+$18, CCG$^+$20, HHPV18] all satisfy this property, hence, they can be used as input of our compiler.

P2P^3-BC UA protocol. This upper bound is based on the observation that when the broadcast channel is available in the last round, it is possible to upgrade the security of the above SA protocol (the one enhanced with the garbled circuit that we have described in the previous paragraph) to UA with the following simple modification: If an honest party is unable to continue computation during Rounds 1 - 3, she simply broadcasts the signal 'abort' in the last round, which would lead to all honest parties aborting unanimously. (Note that a corrupt party can also choose to broadcast 'abort', this does not violate unanimity as all honest parties would abort in such a case.). This takes care of any inconsistency prior to Round 4. Lastly, an adversary cannot cause inconsistency during Round 4, as we make the parties send all their messages via broadcast in Round 4.

BC^3-P2P SIA protocol. To prove this upper bound, we show that a big class of protocols (i.e., those that admit a simulator that can extract the inputs of the corrupted parties in the first three rounds) that are secure with identifiable abort (which use broadcast in all rounds) can be proven to be secure with selective identifiable abort even if the last round is replaced by peer-to-peer channels. Intuitively, if this is not the case, it means that the adversary can make honest parties obtain inconsistent outputs by sending different versions of the last round message. However, this cannot occur since the output of the protocol must have been fixed before the last round (due to our assumption that the simulator extracts the input in the first three rounds), and since that, if there exists a fourth round that forces honest parties to

[4]The consistency of views with respect to the last round follows from input-independence property of the underlying protocol (elaborated in the relevant technical section).

compute the wrong output, this message could be used and sent in the last broadcast round of the original protocol to force honest parties to output an incorrect value. Finally, we note that the protocol proposed in [CRSW22] admits this special simulator. This observation yields a protocol that realizes any function with selective identifiable abort when the communication resources are BC^3-$P2P$.

1.2.2 Impossibility Results

We propose two main categories of impossibility results. In the first category, we show that UA security is impossible to achieve when the communication in the last round is performed over $P2P$. This shows the tightness of our $P2P^3$-BC UA upper bound, completing the picture for UA security. The second category comprises a set of four impossibility results that show that any broadcast pattern that does not use a broadcast channel in each of the first three rounds cannot achieve SIA. This result implies that any SIA secure protocol must rely on the pattern BC^3-$P2P$, hence our protocol is tight. This completes the picture for SIA security. Since IA is stronger than both UA and SIA, both the categories of impossibilities are applicable to IA as well. In particular, by putting everything together we prove that the pattern BC^4 is indeed minimal for realizing security with IA.

BC^3-P2P UA security. The main idea of this impossibility is to show that any protocol that enjoys security with UA in this setting in the plain model can be turned into a 3-round oblivious transfer (OT) protocol in the plain model. Since the latter is known to be impossible [HV16], such a BC^3-$P2P$ UA protocol cannot exist. The transformation occurs in two-steps: First, we show that the BC^3-$P2P$ UA protocol must be such that it is possible for a set of $n/2$ among the n parties to obtain the output by combining their views at the end of Round 3. Intuitively, this is because it may happen that the only communication an honest party, say P, receives in the last round may be from other honest parties. She may still have to compute the output to maintain unanimity – This is because the last round is over peer-to-peer channels and the adversary may have behaved honestly throughout all the rounds towards her fellow-honest party P' (while behaving honestly only in the first three rounds to P). P' will compute the output due to correctness (from the perspective of P', this was an execution where everyone behaved honestly). This lets us infer that the set of honest parties *together* had enough information about the output at the end of Round 3 itself, as this information sufficed to let P get the output at the end of Round 4. Assuming that there are $n/2$ honest parties, this completes the first step. Next, we show that one can construct a three-round OT protocol, where the receiver P_{R} emulates the role of the above set of $n/2$ parties and the sender P_{S} emulates the role of the remaining set of $n/2$ parties. For this, we define the function computed by the n-party BC^3-$P2P$ UA protocol accordingly; and invoke the above claim (of the first step) and security of this n-party protocol to argue correctness and security of the OT protocol respectively.

SIA security. Here, we give a high-level overview of how we prove that SIA is impossible to achieve when the communication pattern is of the form BC-BC-$P2P$-$P2P$. The impossibility of the other communication patterns follows by similar arguments. Assume by contradiction that there exists a three-party protocol Π that can securely compute any efficiently computable function f with SIA security when the broadcast channel is available only in the first two rounds. We denote the parties running this protocol with P_1, P_2, and P_{out}, and assume that f provides the output only to the party P_{out}. We consider now the following two scenarios.

> **Scenario 1.** $P_1^\star$ is corrupted (we denote the i-th corrupted party with $P_i^\star$), and the other parties are honest. $P_1^\star$ behaves like P_1 would, with the difference that it does not send any

message to P_2 in the third and the fourth round. Further, $P_1^\star$ pretends that it did not receive the third and the fourth round (over the point-to-point channel) messages from P_2.

Scenario 2. This time $P_2^\star$ is corrupted, and the other parties are honest. $P_2^\star$ behaves exactly like P_2, but it does not send any message to P_1 in the third and the fourth round. Further, $P_2^\star$ pretends that it did not receive the third and the fourth round (over the point-to-point channel) messages from P_1.

We note that the two scenarios look identical in the eyes of P_{out}. This is because P_{out} cannot access the $P2P$ channel connecting P_1 and P_2, hence, he cannot detect which of the two parties did not send a message. In particular, P_{out} will not be able to detect who is the corrupted party. By the definition of SIA, if P_{out} cannot identify the corrupted party, then it must be able to output the evaluation of the function f. Equipped with this observation, our proof proceeds in two steps.

1. First, we construct a new three-party protocol Π'. We denote the parties running this protocol with P_1', P_2', and P_{out}'. The party P_1' behaves exactly like $P_1^\star$ described in Scenario 1, and similarly P_2' and P_{out}' behave respectively like P_2' and P_{out}' in Scenario 1. We argue that Π' is secure with SA security. In fact, it suffices for our argument to show that Π' is secure for the following two corruption patterns: (a) when P_1' and P_{out}' are corrupt and when (b) P_2' and P_{out}' are corrupt. We refer to the simulators proving security in these cases as $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SIA}}$ and $\mathcal{S}_{2,\mathsf{out}}^{\mathsf{SIA}}$ respectively.

2. Next, we show an attack that allows an adversary $\mathcal{A}^{\mathsf{SA}}$ corrupting $P_2^{\star\prime}$ and $P_{\mathsf{out}}^{\star\prime}$ in Π' to learn the input of honest P_1'. This step would complete the proof as it contradicts the security of Π' for this corruption setting (which was argued to be secure in the first step). Broadly speaking, we show that this adversary $\mathcal{A}^{\mathsf{SA}}$ is able to get access to all the information that the simulator $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SIA}}$ has (which must exists, as argued in the first step). Intuitively, since the information that $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SIA}}$ has must suffice to 'extract' the input of corrupt P_1' (in order for the simulation to be successful[5]), this allows us to argue that $\mathcal{A}^{\mathsf{SA}}$ can use this information to learn the input of honest P_1'.

Before elaborating on each of the above steps, we make the following useful observation: since P_{out}' is the only party getting the output and the security goal of Π' is SA security, we can assume without loss of generality that in Π' **(a)** P_{out}' does not send any message to the other parties in the last round and **(b)** there is no communication between P_1' and P_2' in the last round.

SA security of Π'. In the first step, one can easily observe that correctness of Π' holds as an honest execution of Π' would result in P_{out}' having a view that is identically distributed to the view of P_{out} at the end of Scenario 1 (which sufficed to compute the correct output). Intuitively, privacy holds as there is less room for attack in Π' as compared to Π, as it involves fewer messages. To formally argue SA security of Π' for the case when $P_2^{\star\prime}$ and $P_{\mathsf{out}}^{\star\prime}$ are corrupt, we construct a simulator $\mathcal{S}_{2,\mathsf{out}}^{\mathsf{SA}}$ for Π'. In particular, we need to argue that the messages of P_1' can still be simulated, despite the fact that it does not send messages to $P_2^{\star\prime}$ in the third and the fourth round. Our simulation strategy works as follows. The simulator $\mathcal{S}_{2,\mathsf{out}}^{\mathsf{SA}}$ for Π' internally runs the SIA simulator $\mathcal{S}_{2,\mathsf{out}}^{\mathsf{SIA}}$ of Π for the case where P_1 is honest (recall that this exists by definition). $\mathcal{S}_{2,\mathsf{out}}^{\mathsf{SA}}$ acts as a proxy between $\mathcal{S}_{2,\mathsf{out}}^{\mathsf{SIA}}$ and the corrupted parties for the first and the

[5]Note that $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SIA}}$ works against an adversary corrupting P_1' and P_{out}', and must therefore be able to extract the input of P_1'.

second round, but upon receiving the third round from $\mathcal{S}^{\mathsf{SIA}}_{2,\mathsf{out}}$ directed to $P^{\star}_2$, $\mathcal{S}^{\mathsf{SA}}_{2,\mathsf{out}}$ blocks this message. At this point, a corrupted $P^{\star\prime}_2$ may or may not send a reply, but what is important to observe is that whatever behavior $P^{\star\prime}_2$ has, $P^{\star}_2$ could have had the same behavior while running Π. Intuitively, $P^{\star\prime}_2$ is always weaker than $P^{\star}_2$. Hence, the security of Π can be used to argue that the input of P'_1 remains protected.

We deal with the case where P'_1 and P'_{out} are corrupted in Π' in a similar way. We refer to the technical part of the paper for a more detailed discussion.

Attack by $\mathcal{A}^{\mathsf{SA}}$. In the second step, our goal is to show an adversary $\mathcal{A}^{\mathsf{SA}}$ that corrupts $P^{\star\prime}_2$ and $P^{\star\prime}_{\mathsf{out}}$ and runs the simulator $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ to extract the input of the honest P'_1[6] (proofs with a similar spirit have been considered in [GK96, KO04]). To make the proof go through, we need to argue that an adversary that runs $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ is a *legit adversary*. In particular, this adversary must not rewind the honest P'_1. Note that in the plain model and dishonest majority setting, the only additional power the black-box simulator has compared to an adversary is to perform rewinds. We show that no matter what rewinds $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ performs, these rewinds do not affect the honest party P'_1. At a very high level, $\mathcal{A}^{\mathsf{SA}}$ is able to obtain the same information as $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ would collect over the rewinds because (a) the rewinds that allow $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ to obtain new messages from $P^{\star\prime}_{\mathsf{out}}$ can be locally computed by $\mathcal{A}^{\mathsf{SA}}$ (as $\mathcal{A}^{\mathsf{SA}}$ also controls $P^{\star\prime}_{\mathsf{out}}$) (b) essentially, no rewinds help to obtain new messages from $P^{\star\prime}_1$ because $P^{\star\prime}_1$ does not send any messages to P'_2 (on whose behalf $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ acts) in the last two rounds. In more detail,

Rewinding the second round: $\mathbf{P'_2 \to P'_1}$. Changing the second message may influence the third round that will be computed by P'_1. However, note that P'_1 does not send any message in the third round to P'_2. Hence, we just need to forward to P'_1 only one of the potential multiple second-round messages the simulator generates. The messages we choose to forward need to be picked with some care. We refer the reader to the technical section for more detail.

Rewinding the second round: $\mathbf{P'_2 \to P^{\star\prime}_{out}}$. Changing the second round messages may affect the third round that goes from $P^{\star\prime}_{\mathsf{out}}$ to $P^{\star\prime}_1$, and as such, it may affect the fourth round that goes back from P'_1 to $P^{\star\prime}_{\mathsf{out}}$. However, the simulator $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ acting on behalf of P'_2 will not see the effect of this rewind, given that in Π', P'_2 does not receive any message in the fourth round. We also note that this rewind would additionally allow $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ to obtain new third round messages from $P^{\star\prime}_{\mathsf{out}}$ based on different second round messages of P'_2. However, this can be locally computed by $\mathcal{A}^{\mathsf{SA}}$ in its head, as it controls both $P^{\star\prime}_{\mathsf{out}}$ and $P^{\star\prime}_2$.

The above arguments can be easily extended to infer that any rewind performed in the third round does not affect P'_1. There is one pattern left, which is the one where the simulator rewinds the first round.

Rewinding the first round: $\mathbf{P'_2 \to P'_1}$. The high-level intuition to argue that the simulator has no advantage in using these rewinds is that $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ must be able to work even against the following adversary. Consider a corrupted $P^{\star\prime}_1$ who is rushing in the first round and computes fresh input (and randomness) by evaluating a pseudo-random function (PRF) on the incoming first-round message from P'_2. Subsequently, the corrupted $P^{\star\prime}_1$ uses this input honestly throughout the protocol. It is clear that against such an adversary, a simulator that rewinds the first round has no advantage. This is because changing the first round

[6]There are functionalities for which the simulator may not need to extract any input from the adversary. In our impossibility, we will consider a three-party oblivious transfer functionality (where one party does not have the input), where the simulator must be able to extract the input of the corrupted parties.

would change the input the adversary uses on behalf of $P_1^{\star\prime}$. Therefore, the information collected across the rewinding sessions cannot help to extract the input used by the adversary in the simulated thread (which refers to the transcript that is included in the adversary's view output by the simulator).

Formalizing the above intuition requires some care, and here we provide a slightly more detailed overview of how we do that. Our adversary $\mathcal{A}^{\mathsf{SA}}$ will receive messages of P_1'. $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$ (which we recall is run internally by $\mathcal{A}^{\mathsf{SA}}$) may rewind the first round multiple times, and each time $\mathcal{A}^{\mathsf{SA}}$ must reply with a valid first and second round of P_1'. We could simply reply to $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$ every time using the first round message we received from the honest P_1'. We then forward the first round received from P'_{out} and P_2' to P_1'. P_1' now will send the second round, which we can forward to $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$. Now, $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$ may decide to rewind P_1', by sending a new first round. At this point, we would need to forward this message to P_1', as this is the only way to compute a valid second round of P_1'. Clearly, P_1' is not supposed to reply to such queries, and as such, our adversary $\mathcal{A}^{\mathsf{SA}}$ is stuck. To avoid this problem, we adopt the following strategy. Let us assume that we know in advance that the simulator $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$ runs for at most κ steps[7]. This means that the simulator can open a new session (i.e., rewind the first round) up to κ times. Our adversary samples a random value $i \in [\kappa]$, and for all the sessions $j \neq i$, the adversary will compute the messages on behalf of P_1' using input and randomness computed by evaluating the PRF on input the messages received from P_2'. Only for the i-th session, the adversary will act as a proxy between the messages of P_1' and the simulator $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$. If the $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$ returns a simulated transcript consistent with the i-th session, then we also know that the simulator must have queried the ideal functionality with a value that corresponds to the input of P_1'. Given that we can guess the index i with some non-negligible probability, and given that the simulator will succeed with non-negligible probability as well, our attack would be successful. There is still subtlety though. In the session with indices $j \neq i$, $\mathcal{A}^{\mathsf{SA}}$ internally runs the algorithm of P_1' using an input x_1 that is computed by evaluating a PRF on input the messages generated from P_2'. The input used by the honest P_1' may have a different distribution, and as such, the simulator may decide to never complete the simulation of the i-th session. We first note that, formally, the goal of our adversary $\mathcal{A}^{\mathsf{SA}}$ is not really to extract the input of the honest P_1'. But it is about distinguishing whether the messages that it will receive on behalf of P_1' are generated using the honest procedure of P_1', or using the simulated procedure. Note that in such an MPC security game, the adversary knows, and can actually decide[8] what are the inputs of the honest parties (i.e., what inputs the challenger of the security game will use to compute the messages of P_1'). $\mathcal{A}^{\mathsf{SA}}$ then can internally run $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$, and when the i-th session comes generate an input x_1 by evaluating the PRF on the messages received on the behalf of P_2'. Now that the input of the honest P_1' is defined, we start the indistinguishability game with a challenger that takes as input x_1 (and some default input for the corrupted parties). In this way, we have the guarantee that when the challenger is not generating simulated messages, all the sessions look identical in the eyes of the simulator $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$. Hence, we can correctly state that with some non-negligible probability, it will return a simulated transcript for the i-th session. Note that $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$ will return $\tilde{x}_1$ when querying the ideal functionality in the i-th session, and we will have that $\tilde{x}_1 = x_1$ iff the challenger is computing the messages using the honest procedure of P_1'. If instead, the challenger was generating simulated messages

[7] If the simulator has expected polynomial time κ, for some polynomial κ, then our adversary will run the simulator up to κ steps. This will guarantee that the simulator will terminate successfully with some non-negligible probability.

[8] The security of MPC states that security holds for any honest parties' inputs (decided before the experiment starts), and these inputs may be known to the adversary.

on behalf of P_1', then the probability that $\tilde{x}_1 = x_1$ is small[9]. Hence, this will give a non-negligible advantage to $\mathcal{A}^{\mathsf{SA}}$ in distinguishing what the MPC challenger is doing. We refer to the technical sections of the paper for a more formal treatment of this proof.

Rewinding the first round: $\mathbf{P_2'} \rightarrow \mathbf{P_{out}^{\star\prime}}$. To argue this case, we note that if $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$ acts against the rushing adversary defined in the above case (where $P_1^{\star\prime}$ changes its input based on the output of PRF applied on the first round message from P_2'), then the first and second round messages of $P_1^{\star\prime}$ obtained during the rewinds can be locally emulated by $\mathcal{A}^{\mathsf{SA}}$ (as he controls both $P_2^{\star\prime}$ and $P_{\mathsf{out}}^{\star\prime}$).

In summary, we have argued that $\mathcal{A}^{\mathsf{SA}}$ can internally run the simulator $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$ which enables the adversary to be able to extract the input of P_1'.[10] We refer to the technical section of the paper for a much more formal proof, and for the proof of impossibility results related to the other communication patterns.

1.3 Related Work

The work of [CGZ20] initiated the study of broadcast-optimal MPC. They investigated the question of the best security guarantees that can be achieved by all possible broadcast patterns in two-round secure computation protocols, namely no broadcast, broadcast (only) in the first round, broadcast (only) in the second round, and broadcast in both rounds. Their results focused on the dishonest majority setting and assumed a setup (such as PKI or CRS) [11]. The works of [DMR+21, DRSY23] investigate the same question for two-round MPC with setup (such as PKI or CRS), but in the honest-majority setting. We refer the reader to [CGZ20, DMR+21, DRSY23] for a detailed overview of the state of the art on 2-round MPC and their use of broadcast. The work of [GJPR21] studies the best achievable security for two-round MPC in the plain model for different communication models such as only broadcast channels, only peer-to-peer channels, or both. Unlike the previously mentioned line of work, this work does not consider communication patterns where broadcast is limited to one of the two rounds. Going beyond two rounds, the work of [BMMR21] studies broadcast-optimal three-round MPC with guaranteed output delivery given an honest majority and CRS, and shows that the use of broadcast in the first two rounds is necessary. None of the above works consider the dishonest majority setting without setup (i.e. the plain model). In this setting, there are several existing round-optimal (four round) constructions, namely protocols with unanimous abort in [ACJ17, BGJ+18, BHP17, CCG+20, HHPV18] and with identifiable abort in [CRSW22]. However, these works do not restrict the use of broadcast in any round. To the best of our knowledge, we are the first to investigate the question of optimizing broadcast for round optimal (four-round) protocols in the dishonest majority setting without setup (i.e. in the plain model).

2 Preliminaries

2.1 Pseudorandom Function

We use the definition of the pseudorandom function (PRF) from [KL14]

[9] This will depend on the domain size of P_1' input and on the type of function we are computing.

[10] The simulator may be expected polynomial time, hence we need to cut the running time of the simulator to make sure that $\mathcal{A}^{\mathsf{SA}}$ remains PPT.

[11] It is necessary to assume setup for two-round protocols in dishonest majority setting.

Definition 1 (Pseudorandom function)**.** *Let* $\mathtt{F} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ *be an efficient, length-preserving keyed function.* $\mathtt{F}$ *is a pseudorandom function if for all PPT distinguisher* $\mathcal{D}$*, there is a negligible function negl s.t.:*

$$\left| \Pr[\mathcal{D}^{\mathtt{F}_k(\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{D}^{f(\cdot)}(1^\lambda) = 1] \right|$$

where the first probability is taken over uniformly chosen of $k \in \{0,1\}^\lambda$ *and the randomness of* $\mathcal{D}$*, and the second probability is taken over uniformly chosen truly random function* f*, and the randomness of* $\mathcal{D}$

2.2 Garbling scheme

We use the definition of the garbling scheme from [CGZ20].

Definition 2 (Garbling scheme [BHR12, CGZ20, DMR+21])**.** *A projective garbling scheme is a pair of algorithms* (garble, eval)*, such that:*

- $(\mathtt{GC}, \mathbf{K}) \leftarrow \mathtt{garble}(1^\lambda, \mathtt{C})$*: on input the unary representation of the security parameter* 1^λ *and a boolean circuit* $\mathtt{C} : \{0,1\}^{\mathsf{L}} \to \{0,1\}^m$*, the garbling algorithm outputs a garbled circuit* $\mathtt{GC}$ *and* L *pairs of garbled labels* $\mathbf{K} = \{\mathbf{K}_\alpha^b\}_{b\in\{0,1\},\alpha\in[\mathsf{L}]}$*. For simplicity, we assume that for every* $\alpha \in [\mathsf{L}]$ *and* $b \in \{0,1\}$*, it holds that* $\mathbf{K}_\alpha^b \in \{0,1\}^\lambda$*.*
- $y \leftarrow \mathtt{eval}(\mathtt{GC}, \{\mathbf{K}_\alpha\}_{\alpha\in[\mathsf{L}]})$*: on input a garbled circuit* $\mathtt{GC}$ *and* L *garbled labels* $\{\mathbf{K}_\alpha\}_{\alpha\in[\mathsf{L}]}$*, the evaluation algorithm output a value* $y \in \{0,1\}^m$*.*

The scheme has the following two properties:

- Correctness*: For any boolean circuit* $\mathtt{C} : \{0,1\}^{\mathsf{L}} \to \{0,1\}^m$*, and* $x = (x_1, \ldots, x_{\mathsf{L}}) \in \{0,1\}^{\mathsf{L}}$*, it holds that:*

  $$\Pr[\mathtt{eval}(\mathtt{GC}, \mathbf{K}[x]) \neq \mathtt{C}(x)] \leq negl(\lambda)$$

  where $(\mathtt{GC}, \mathbf{K}) \leftarrow \mathtt{garble}(1^\lambda, \mathtt{C})$*, with* $\mathbf{K} = (\mathbf{K}_1^0, \mathbf{K}_1^1, \ldots, \mathbf{K}_{\mathsf{L}}^0, \mathbf{K}_{\mathsf{L}}^1)$ *and* $\mathbf{K}[x] = (\mathbf{K}_1^{x_1}, \ldots, \mathbf{K}_{\mathsf{L}}^{x_{\mathsf{L}}})$*.*
- (Adaptive) Privacy*: There exists a probabilistic polynomial-time (PPT) simulator* $\mathtt{simGC}$*, s.t. for every PPT adversary* $\mathcal{A}$*:*

  $$\left| \Pr\left[\mathtt{Expt}_{\Pi,\mathcal{A},\mathtt{simGC}}(1^\lambda, 0) = 1\right] - \Pr\left[\mathtt{Expt}_{\Pi,\mathcal{A},\mathtt{simGC}}(1^\lambda, 1) = 1\right] \right| \leq negl(\lambda)$$

  where the experiment $\mathtt{Expt}_{\Pi,\mathcal{A},\mathtt{simGC}}(1^\lambda, b)$ *is defined as follows:*

  - *The adversary* $\mathcal{A}$ *specify the boolean circuit* $\mathtt{C} : \{0,1\}^{\mathsf{L}} \to \{0,1\}^m$*.*
  - *If* $b = 0$*, the challenger* $\mathcal{C}$ *computes* $(\mathtt{GC}, \mathbf{K}) \leftarrow \mathtt{garble}(1^\lambda, \mathtt{C})$*, and returns* $\mathtt{GC}$*.*
  - *If* $b = 1$*, the challenger* $\mathcal{C}$ *computes* $\mathtt{GC} \leftarrow \mathtt{simGC}(1^\lambda, \phi(\mathtt{C}), \text{“}ckt\text{”})$*, where* $\phi(\mathtt{C})$ *denotes the topology of* $\mathtt{C}$[12]*. The challenger returns* $\mathtt{GC}$*.*
  - *The adversary* $\mathcal{A}$ *specify input* $x = (x_1, \ldots, x_{\mathsf{L}}) \in \{0,1\}^{\mathsf{L}}$*.*
  - *If* $b = 0$*, the challenger* $\mathcal{C}$ *sets* $\mathbf{K}_\alpha = \mathbf{K}_\alpha^{x_\alpha}$*, for* $\alpha \in \mathsf{L}$ *and returns* $\{\mathbf{K}_\alpha\}_{\alpha\in\mathsf{L}}$*.*

[12] We assume that the topology of a circuit does not reveal hard coded values (as hard-coded values are essentially fixed input labels for some wires.)

– *If* $b = 1$*, computes* $\mathbf{K}_1, \ldots, \mathbf{K}_{\mathsf{L}} \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}(x), \text{“}input\text{”})$*, and returns* $\{\mathbf{K}_\alpha\}_{\alpha \in \mathsf{L}}$*.*
– $\mathcal{A}$ *outputs a bit* b'*, and it is the output of the experiment.*

- Partial Evaluation Resiliency [DMR+21]: *We say that* GC *satisfies* partial evaluation resiliency *if there exists a simulator* simGC *such that for every PPT adversary* $\mathcal{A}$*, it holds that*

$$\left| \Pr\left[\mathtt{Expt}^{\mathsf{PR}}_{\Pi,\mathcal{A},\mathtt{simGC}}(1^\lambda, 0) = 1\right] - \Pr\left[\mathtt{Expt}^{\mathsf{PR}}_{\Pi,\mathcal{A},\mathtt{simGC}}(1^\lambda, 1) = 1\right] \right| \leq negl(\lambda)$$

where the experiment $\mathtt{Expt}^{\mathsf{PR}}_{\Pi,\mathcal{A},\mathtt{simGC}}(1^\lambda, b)$ *is defined as follows:*

– *The adversary* $\mathcal{A}$ *specify the boolean* $\mathtt{C} : \{0,1\}^{\mathsf{L}} \to \{0,1\}^m$*,* $i \in \mathsf{L}$*, and* $x = (x_1, \ldots, x_{\mathsf{L}}) \in \{0,1\}^{\mathsf{L}}$
– *If* $b = 0$*, the challenger* $\mathcal{C}$ *computes* $(\mathtt{GC}, \mathbf{K}_1^{x_1}, \ldots, \mathbf{K}_{\mathsf{L}}^{x_{\mathsf{L}}}) \leftarrow \mathtt{garble}(1^\lambda, \mathtt{C})$*, and returns* $(\mathtt{GC}, \{\mathbf{K}_\alpha^{x_\alpha}\}_{\alpha \in \mathsf{L}, \alpha \neq i})$
– *If* $b = 1$*, the challenger* $\mathcal{C}$ *computes* $(\mathtt{GC}, \{\mathbf{K}_\alpha\}_{\alpha \in \mathsf{L}}) \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}, \mathtt{C}(x))$*, and returns* $(\mathtt{GC}, \{\mathbf{K}_\alpha\}_{\alpha \in \mathsf{L}, \alpha \neq i})$
– $\mathcal{A}$ *outputs a bit* b'*, and it is the output of the experiment.*

Instantiation. As shown by Bellare *et al.* [BHR12], adaptive garbled circuits can be obtained using Yao's garbled circuits and one-time pads. We refer to [DMR+21] for details on how any garbling scheme can be augmented using one-time pads to satisfy partial evaluation resiliency.

2.3 Additive secret sharing scheme

In our work, we will use *n-out-of-n secret sharing scheme*, which is also referred to as *additive secret sharing scheme*. We give the definition here:

Definition 3 (n-out-of-n secret sharing scheme)**.** *A n-out-of-n secret sharing scheme is a pair of algorithms (*share*,* reconstruct*), such that:*

- share *is a randomized algorithm that on input a secret* m*, outputs a set of* n *shares* $(s_1, \ldots, s_n)$*.*
- reconstruct *is a deterministic algorithm that on input* n *shares* $(s_1, \ldots, s_n)$*, outputs the secret* m*.*

The scheme needs to satisfy the correctness *property if : For any* m *we have:*

$$\Pr_{\mathtt{share}(m) \to (s_1, \ldots, s_n)}[\mathtt{reconstruct}(s_1, \ldots, s_n) = m] = 1$$

In addition, the scheme satisfies perfect security *property if: for all* m, m' *the following distributions are indistinguishable:*

$$\{(s_1, \ldots, s_n) \mid (s_1, \ldots, s_n) \leftarrow \mathtt{share}(m)\}$$
$$\{(s'_1, \ldots, s'_n) \mid (s'_1, \ldots, s'_n) \leftarrow \mathtt{share}(m')\}$$

3 Secure Multiparty Computation (MPC) Definitions

3.1 Security Model

We follow the real/ideal world simulation paradigm and we adopt the security model of Cohen, Garay and Zikas [CGZ20].

Real-world. An n-party protocol $\Pi = (P_1, \ldots, P_n)$ is an n-tuple of probabilistic polynomial-time (PPT) interactive Turing machines (ITMs), where each party P_i is initialized with input $x_i \in \{0,1\}^*$ and random coins $r_i \in \{0,1\}^*$. We let $\mathcal{A}$ denote a special PPT ITM that represents the adversary and that is initialized with input that contains the identities of the corrupt parties, their respective private inputs, and an auxiliary input.

The protocol is executed in rounds (i.e., the protocol is synchronous). Each round consists of the send phase and the receive phase, where parties can respectively send the messages from this round to other parties and receive messages from other parties. In every round parties can communicate either over a broadcast channel or a fully connected $P2P$ network. We assume that these channels are private and we assume the channels to be ideally authenticated.

During the execution of the protocol, the corrupt parties receive arbitrary instructions from the adversary $\mathcal{A}$, while the honest parties faithfully follow the instructions of the protocol. We consider the adversary $\mathcal{A}$ to be rushing, i.e., during every round the adversary can see the messages the honest parties sent before producing messages from corrupt parties.

At the end of the protocol execution, the honest parties produce output, and the adversary outputs an arbitrary function of the corrupt parties' view. The view of a party during the execution consists of its input, random coins and the messages it sees during the execution.

Definition 4 (Real-world execution)**.** *Let $\Pi = (P_1, \ldots, P_n)$ be an n-party protocol and let $\mathcal{I} \subseteq [n]$, of size at most t, 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 world, on input vector $x = (x_1, \ldots, x_n)$, auxiliary input aux and the unary representation of the security parameter 1^λ, denoted $\mathtt{REAL}_{\Pi,\mathcal{I},\mathcal{A}(\mathsf{aux})}(x, 1^\lambda)$, is defined as the output vector of $P_1, \ldots, P_n$ and $\mathcal{A}(\mathsf{aux})$ resulting from the protocol interaction.*

Ideal-world. We describe ideal world executions with selective abort (sa-abort), selective identifiable abort (si-abort), unanimous abort (un-abort), identifiable abort (id-abort).

Definition 5 (Ideal Computation)**.** *Consider $\mathsf{type} \in \{\mathsf{sa\text{-}abort}, \mathsf{un\text{-}abort}, \mathsf{si\text{-}abort}, \mathsf{id\text{-}abort}\}$. Let $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be an n-party function and let $\mathcal{I} \subseteq [n]$, of size at most t, be the set of indices of the corrupt parties. Then, the joint ideal execution of f under $(\mathcal{S}, \mathcal{I})$ on input vector $x = (x_1, \ldots, x_n)$, auxiliary input aux to $\mathcal{S}$ and the unary representation of the security parameter 1^λ, denoted $\mathtt{IDEAL}^{\mathsf{type}}_{f,\mathcal{I},\mathcal{S}(\mathsf{aux})}(x, 1^\lambda)$, is defined as the output vector of $P_1, \ldots, P_n$ and $\mathcal{S}$ resulting from the following ideal process.*

1. Parties send inputs to trusted party*: An honest party P_i sends its input x_i to the trusted party. The simulator $\mathcal{S}$ may send to the trusted party arbitrary inputs for the corrupt parties. Let x'_i be the value actually sent as the input of party P_i.*
2. Trusted party speaks to simulator*: The trusted party computes $(y_1, \ldots, y_n) = f(x'_1, \ldots, x'_n)$. If there are no corrupt parties proceed to step 4.*
   (a) If $\mathsf{type} \in \{\mathsf{sa\text{-}abort}, \mathsf{un\text{-}abort}, \mathsf{si\text{-}abort}, \mathsf{id\text{-}abort}\}$: *The trusted party sends $\{y_i\}_{i \in \mathcal{I}}$ to $\mathcal{S}$.*
3. Simulator $\mathcal{S}$ responds to trusted party*:*
   (a) If $\mathsf{type} = \mathsf{sa\text{-}abort}$: *The simulator $\mathcal{S}$ can select a set of parties that will not get the output as $\mathcal{J} \subseteq [n] \setminus \mathcal{I}$. (Note that $\mathcal{J}$ can be empty, allowing all parties to obtain the output.) It sends $(\mathtt{abort}, \mathcal{J})$ to the trusted party.*
   (b) If $\mathsf{type} = \mathsf{un\text{-}abort}$: *The simulator can send $\mathtt{abort}$ to the trusted party. If it does, we take $\mathcal{J} = [n] \setminus \mathcal{I}$.*

(c) If $\mathsf{type} = \mathsf{si\text{-}abort}$: *The simulator $\mathcal{S}$ can select a set of parties that will not get the output as $\mathcal{J} \subseteq [n] \setminus \mathcal{I}$. (Note that $\mathcal{J}$ can be empty, allowing all parties to obtain the output.) For each party j in $\mathcal{J}$, the adversary selects a corrupt party $i_j^* \in \mathcal{I}$ who will be blamed by party j. It sends $(\mathtt{abort}, \mathcal{J}, \{j, i_j^*\}_{j\in\mathcal{J}})$ to the trusted party.*

(d) If $\mathsf{type} = \mathsf{id\text{-}abort}$: *If it chooses to abort, the simulator $\mathcal{S}$ can select a corrupt party $i^* \in \mathcal{I}$ who will be blamed, and send $(\mathtt{abort}, i^*)$ to the trusted party. If it does, we take $\mathcal{J} = [n] \setminus \mathcal{I}$.*

4. Trusted party answers parties*:*

(a) *If the trusted party got* $\mathtt{abort}$ *from the simulator $\mathcal{S}$,*

i. *It sets the abort message* $\mathsf{abortmsg}$*, as follows:*

- *if* $\mathsf{type} \in \{\mathsf{sa\text{-}abort}, \mathsf{un\text{-}abort}\}$*, we let* $\mathsf{abortmsg} = \perp$.
- *if* $\mathsf{type} = \mathsf{si\text{-}abort}$*, we let* $\mathsf{abortmsg} = \{\mathsf{abortmsg}_j\}_{j\in\mathcal{J}} = (\perp, i_j^*)_{j\in\mathcal{J}}$.
- *if* $\mathsf{type} = \mathsf{id\text{-}abort}$*, we let* $\mathsf{abortmsg} = (\perp, i^*)$.

ii. *The trusted party sends y_j to every party P_j, $j \in [n] \setminus \mathcal{J}$.*
If $\mathsf{type} = \mathsf{si\text{-}abort}$*, the trusted party then sends* $\mathsf{abortmsg}_j$ *to each party P_j, $j \in \mathcal{J}$; otherwise, the trusted party sends* $\mathsf{abortmsg}$ *to every party P_j, $j \in \mathcal{J}$*

(b) *Otherwise, it sends y to every P_j, $j \in [n]$.*

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

Security Definitions. We now define the security notion for protocols.

Definition 6. *Consider* $\mathsf{type} \in \{\mathsf{sa\text{-}abort}, \mathsf{un\text{-}abort}, \mathsf{si\text{-}abort}, \mathsf{id\text{-}abort}\}$. *Let $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be an n-party function. A protocol Π t-securely computes the function f with* type *security if for every PPT real-world adversary $\mathcal{A}$ with auxiliary input* aux*, there exists a PPT simulator $\mathcal{S}$ such that for every $\mathcal{I} \subseteq [n]$ of size at most t, for all $x \in (\{0,1\}^*)^n$, for all $\lambda \in \mathbb{N}$, it holds that*

$$\mathtt{REAL}_{\Pi,\mathcal{I},\mathcal{A}(\mathsf{aux})}(x, 1^\lambda) \stackrel{c}{\equiv} \mathtt{IDEAL}^{\mathsf{type}}_{f,\mathcal{I},\mathcal{S}(\mathsf{aux})}(x, 1^\lambda).$$

3.2 Notation

In this paper, we mainly focus on four-round secure computation protocols. Rather than viewing a protocol Π as an n-tuple of interactive Turing machines, it is convenient to view each Turing machine as a sequence of multiple algorithms: $\mathtt{frst\text{-}msg}_i$, to compute P_i's first messages to its peers; $\mathtt{nxt\text{-}msg}_i^k$, to compute P_i's $(k+1)$-th round messages for $(1 \le k \le 3)$; and $\mathtt{output}_i$, to compute P_i's output. Thus, a protocol Π can be defined as $\{(\mathtt{frst\text{-}msg}_i, \mathtt{nxt\text{-}msg}_i^k, \mathtt{output}_i)\}_{i\in[n],k\in\{1,2,3\}}$.

The syntax of the algorithms is as follows:

- $\mathtt{frst\text{-}msg}_i(x_i; r_i) \to (\mathsf{msg}^1_{i\to 1}, \ldots, \mathsf{msg}^1_{i\to n})$ produces the first-round messages of party P_i to all parties. Note that a party's message to itself can be considered to be its state.
- $\mathtt{nxt\text{-}msg}_i^k(x_i, \{\mathsf{msg}^l_{j\to i}\}_{j\in[n],l\in\{1,2,\ldots,k\}}; r_i) \to (\mathsf{msg}^{k+1}_{i\to 1}, \ldots, \mathsf{msg}^{k+1}_{i\to n})$ produces the $(k+1)$-th round messages of party P_i to all parties.
- $\mathtt{output}_i(x_i, \mathsf{msg}^1_{1\to i}, \ldots, \mathsf{msg}^1_{n\to i}, \ldots, \mathsf{msg}^j_{1\to i}, \ldots, \mathsf{msg}^j_{n\to i}; r_i) \to y_i$ produces the output returned to party P_i.

When the first round is over broadcast channels, we consider $\texttt{frst-msg}_i$ to return only one message — msg_i^1. Similarly, when the $(k+1)$-th round is over broadcast channels, we consider $\texttt{nxt-msg}_i^k$ to return only msg_i^{k+1}. We also note that, unless needed, to not overburden the notation, we do not pass the random coin r as an explicit input of the cryptographic algorithms. We denote " $\leftarrow$ " as the assigning operator (e.g. to assign to a the value of b we write $a \leftarrow b$). We denote the acronym BC to indicate a round where broadcast is available and the acronym $P2P$ to indicate a round where only peer-to-peer channels are available. We use the notation $P2P^x$ (BC^x) to indicate x rounds of peer-to-peer (broadcast) communications. To strengthen our results, our lower bounds assume that the BC rounds allow peer-to-peer communication as well; our upper bounds assume that the BC rounds involve only broadcast messages (and no peer-to-peer messages).

4 Positive Results

4.1 P2P^4, SA, Plain Model, n > t

In this section, we want to demonstrate that it is feasible to construct a 4-round protocol with SA security, in order to do so we show a compiler that on input a 4-round protocol Π_{bc} with unanimous abort which makes use of the broadcast channel in the dishonest majority setting gives us a 4-round protocol $\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4}$ with the same threshold corruption for selective abort, but relying only on $P2P$ communication. Further, we assume that the exists a simulator for Π_{bc} which extracts the inputs of the adversary from the first three rounds. For instance, one can instantiate Π_{bc} using the protocol of [CCG$^+$20][13].

At a very high level, our compiler follows the approach of Cohen *et al.* [CGZ20]. The approach of Cohen *et al.* focuses on the 2-round setting (using some form of setup) and compiles a 2-round protocol Π_{bc} which uses broadcast in both rounds into one that works over peer-to-peer channels. This core idea of the compiler is to guarantee that honest parties have the same view of the first-round message when they need to compute their second-round message. To achieve this goal the parties, in the first round, generate a garbled circuit which computes their second-round message of Π_{bc} and they secret share their labels using additive secret sharing. The parties send the first-round message of Π_{bc}. In the second round, each party sends her garbled circuit and for each received first-round message of Π_{bc} she sends her appropriate share corresponding to the label in everyone else's garbled circuit. The important observation is that the labels are reconstructed only when parties send the same first-round message to *every* other party. In this work, we extend the following approach for four rounds executing the above idea for Rounds 1 - 2 and subsequently for Rounds 3 - 4. If at any round a party detects any inconsistency (e.g., the garbled circuit outputs $\perp$, or she did not receive a message from another party) she simply aborts. Moreover, the protocol requires some changes w.r.t. the original approach since a corrupted party can send (in the second round) different garbled circuits to the honest party obtaining different 2nd rounds of Π_{bc}. We need to ensure that honest parties abort if the adversary does so, to guarantee that the adversary does not obtain honest parties' responses computed with respect to different versions, in the subsequent rounds. Therefore, the garbled circuits are sent in the round that they are generated in and echoed in the next round.

In more detail, the security follows from the security of Π_{bc} because of the following: the only advantage the adversary has in comparison to Π_{bc} is that she can send inconsistent first (resp., third-round messages) over $P2P$ channels. However, additive sharing of the labels of

[13]To the best of our knowledge, simulators of all existing 4-round construction in the plain model (e.g., [BGJ$^+$18, CCG$^+$20, HHPV18]) have this property of input extraction before the last round. In particular, see page 42 of [CCG$^+$19] for details regarding input extraction by the simulator of the UA protocol in [CCG$^+$20].

the honest party's garbled circuit ensures that the adversary can obtain second round (resp., fourth-round) of an honest party only if she sent identical first-round (resp., third-round) to *all* honest parties. Therefore, if the honest parties do not abort, it must be the case that they have a consistent view with respect to the first and third-round messages of the adversary. Further, since the honest parties also echo the garbled circuits sent by the adversary (computing the corrupt parties' second-round messages), if they proceed to evaluate those, it would mean that the honest parties are agreeing with respect to the second-round messages of the adversary. Note that this does not constitute an issue in the 4th round. If the adversary manages to send garbled circuits resulting in honest parties obtaining different valid fourth rounds of $\Pi_{\mathtt{bc}}$ that result in different outputs, this would violate the security of $\Pi_{\mathtt{bc}}$. This follows from our assumption that the simulator of $\Pi_{\mathtt{bc}}$ extracts the input of the adversary in the first three rounds, which guarantees that the adversarial inputs of $\Pi_{\mathtt{bc}}$ are fixed before the last round. Intuitively, in the last round of $\Pi_{\mathtt{bc}}$, the adversary can only decide if the honest parties obtain the output or not. Finally, it is important to observe that the compiler avoids using zero-knowledge proofs (as any misbehavior that the adversary does such as garbling an incorrect function can be translated to the adversary broadcasting the corresponding second and fourth-round message in the underlying protocol $\Pi_{\mathtt{bc}}$) and uses only tools that can be instantiated from one-way functions.

In Figure 4.1 we formally describe our protocol $\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4}$.

Figure 4.1: $\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4}$

Primitives: A four-broadcast-round protocol $\Pi_{\mathtt{bc}}$ that securely computes f with unanimous abort security against $t < n$ corruptions, and a garbling scheme $(\mathtt{garble}, \mathtt{eval}, \mathtt{simGC})$. For simplicity assume that each round message has the same length and it is ℓ bits long, so each circuit has $\mathsf{L} = n \cdot \ell$ input bits.
Notation: Let $\mathtt{C}^j_{i,x}(\mathsf{msg}^j_1, \ldots, \mathsf{msg}^j_n)$ denote the boolean circuit with hard-wired values x that takes as input the j-th round messages $\mathsf{msg}^j_1, \ldots, \mathsf{msg}^j_n$ and computes $\mathtt{nxt\text{-}msg}^j_i$. We assume that when a party aborts she also signals the abort to all other parties.
Private input: Every party P_i has a private input $x_i \in \{0,1\}^*$.

First round (P2P): Every party P_i does the following:

1. Let $\mathsf{msg}^1_i \leftarrow \mathtt{frst\text{-}msg}_i(x_i)$ be P_i's first round message in $\Pi_{\mathtt{bc}}$.
2. Compute $(\mathtt{GC}_i, \mathbf{K}_i) \leftarrow \mathtt{garble}(1^\lambda, \mathtt{C}^1_{i,x_i})$, where $\mathbf{K}_i = \{\mathbf{K}^b_{i,\alpha}\}_{\alpha\in[\mathsf{L}],b\in\{0,1\}}$.
3. For every $\alpha \in [\mathsf{L}]$ and $b \in \{0,1\}$, sample n uniform random strings $\{\mathbf{K}^b_{i\to k,\alpha}\}_{k\in[n]}$, such that $\mathbf{K}^b_{i,\alpha} = \bigoplus_{k\in[n]} \mathbf{K}^b_{i\to k,\alpha}$.
4. Send to every party P_j the message $(\mathsf{msg}^1_i, \mathtt{GC}_i, \{\mathbf{K}^b_{i\to j,\alpha}\}_{\alpha\in[\mathsf{L}],b\in\{0,1\}})$

Second round (P2P): Every party P_i does the following:

1. If P_i does not receive a message from some other party (or an abort message), she aborts;
2. Otherwise, let $(\mathsf{msg}^1_{j\to i}, \mathtt{GC}_i, \{\mathbf{K}^b_{j\to i,\alpha}\}_{\alpha\in[\mathsf{L}],b\in\{0,1\}})$ be the first round message received from P_j.
3. Concatenate all received messages $\{\mathsf{msg}^1_{j\to i}\}_{j\in[n]}$ as $(\mu^1_{i,1}, \ldots, \mu^1_{i,\mathsf{L}}) \leftarrow (\mathsf{msg}^1_{1\to i}, \ldots, \mathsf{msg}^1_{n\to i}) \in \{0,1\}^{\mathsf{L}}$.

4. Let $\overline{\mathsf{GC}}_i$ be the set of garbled circuits received from the other parties in the first round.

5. Send to all parties the message $(\overline{\mathsf{GC}}_i, \{\mathbf{K}_{j\to i,\alpha}^{\mu^1_{i,\alpha}}\}_{j\in[n],\alpha\in[\mathsf{L}]})$.

Third round (P2P): Every party P_i does the following:

1. If P_i does not receive a message from some other party (or receives an abort message), she aborts; Otherwise, let $(\{\overline{\mathsf{GC}}_l\}_{l\in[n]}, \{\mathbf{K}_{1\to j,\alpha}\}_{\alpha\in[\mathsf{L}]}, \ldots, \{\mathbf{K}_{n\to j,\alpha}\}_{\alpha\in[\mathsf{L}]})$ be the second round message received from party P_j, and let $\mathtt{GC}_j$ be the garbled circuit received from P_j in the first round.

2. Check if the set of garbled circuits $\{\overline{\mathsf{GC}}_l\}_{l\in[n]}$ echoed in Round 2 are consistent with the garbled circuits received in Round 1. If this is not the case, abort.

3. For every $j \in [n]$ and $\alpha \in [\mathsf{L}]$, reconstruct each garbled label by computing $\mathbf{K}_{j,\alpha} \leftarrow \bigoplus_{k\in[n]} \mathbf{K}_{j\to k,\alpha}$.

4. For every $j \in [n]$, evaluate the garble circuit as $\mathsf{msg}_j^2 \leftarrow \mathtt{eval}(\mathtt{GC}_j, \{\mathbf{K}_{j,\alpha}\}_{\alpha\in[\mathsf{L}]})$. If any evaluation fails, aborts. Let $\mathsf{msg}_i^3 \leftarrow \mathtt{nxt\text{-}msg}_i^2(x_i, \{\mathsf{msg}_{j\to i}^1\}_{j\in[n]}, \{\mathsf{msg}_j^2\}_{j\in[n]})$ be the P_i's third round message in Π_{bc}.

5. Compute $(\widetilde{\mathtt{GC}}_i, \widetilde{\mathbf{K}}_i) \leftarrow \mathtt{garble}(1^\lambda, \mathtt{C}_{i,x_i}^3)$, where $\widetilde{\mathbf{K}}_i = \{\widetilde{\mathbf{K}}_{i,\alpha}^b\}_{\alpha\in[\mathsf{L}],b\in\{0,1\}}$.

6. For every $\alpha \in [\mathsf{L}]$ and $b \in \{0,1\}$, sample n uniform random strings $\{\widetilde{\mathbf{K}}_{i\to j,\alpha}^b\}_{j\in[n]}$, such that $\widetilde{\mathbf{K}}_{i,\alpha}^b = \bigoplus_{k\in[n]} \widetilde{\mathbf{K}}_{i\to k,\alpha}^b$.

7. Send to every party P_j the message $(\mathsf{msg}_i^3, \{\widetilde{\mathbf{K}}_{i\to j,\alpha}^b\}_{\alpha\in[\mathsf{L}],b\in\{0,1\}})$

Fourth round (P2P): Every party P_i does the following:

1. If P_i does not receive a message from some other party (or receives an abort message), she aborts;

2. Otherwise, let $(\mathsf{msg}_{j\to i}^3, \{\widetilde{\mathbf{K}}_{j\to i,\alpha}^b\}_{\alpha\in[\mathsf{L}],b\in\{0,1\}})$ be the third round message received from P_j.

3. Concatenate all received messages $\{\mathsf{msg}_{j\to i}^3\}_{j\in[n]}$ as $(\mu_{i,1}^2, \ldots, \mu_{i,\mathsf{L}}^2) \leftarrow (\mathsf{msg}_{1\to i}^3, \ldots, \mathsf{msg}_{n\to i}^3) \in \{0,1\}^{\mathsf{L}}$

4. Send to all parties the message $(\widetilde{\mathtt{GC}}_i, \{\widetilde{\mathbf{K}}_{j\to i,\alpha}^{\mu^2_{i,\alpha}}\}_{j\in[n],\alpha\in[\mathsf{L}]})$

Output Computation: Every party P_i does the following:

1. If P_i does not receive a message from some other party (or receives an abort message), she aborts; Otherwise, let $(\widetilde{\mathtt{GC}}_j, \{\widetilde{\mathbf{K}}_{1\to j,\alpha}\}_{\alpha\in[\mathsf{L}]}, \ldots, \{\widetilde{\mathbf{K}}_{n\to j,\alpha}\}_{\alpha\in[\mathsf{L}]})$ be the fourth round message received from party P_j.

2. For every $j \in [n]$ and $\alpha \in [\mathsf{L}]$, reconstruct each garbled label by computing $\widetilde{\mathbf{K}}_{j,\alpha} \leftarrow \bigoplus_{k\in[n]} \widetilde{\mathbf{K}}_{j\to k,\alpha}$

3. For every $j \in [n]$, evaluate the garbled circuits as $\mathsf{msg}_j^4 \leftarrow \mathtt{eval}(\widetilde{\mathtt{GC}}_j, \{\widetilde{\mathbf{K}}_{j,\alpha}\}_{\alpha\in[\mathsf{L}]})$. If any evaluation fails, aborts.

4. Compute and output $y \leftarrow \mathtt{output}_i(x_i, \{\mathsf{msg}_{j\to i}^1\}_{j\in[n]}, \{\mathsf{msg}_j^2\}_{j\in[n]}, \{\mathsf{msg}_{j\to i}^3\}_{j\in[n]}, \{\mathsf{msg}_j^4\}_{j\in[n]})$

Theorem 1 (*P2P-P2P-P2P-P2P*, SA, Plain Model, $n > t$)**.** *Let f be an efficiently computable n-party function, where $n > t$. Let Π_{bc} be a BC-BC-BC-BC protocol that securely computes f with unanimous abort security against $t < n$ corruptions with the additional constraint that a simulator can extract inputs before the last round. Then, assuming secure garbling schemes (Definition 2), the protocol from Figure 4.1 can compute f with selective-abort security that uses only P2P channels against $t < n$ corruptions.*

Proof. We prove the protocol from Figure 4.1 computes f securely with selective abort. The proof follows similarly to arguments lemma 4.9 in [CGZ20], with some modifications since we are focusing on the plain model.

Let $\mathcal{A}$ be the adversary attacking protocol $\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4}$, and let $\mathcal{I} \subseteq [n]$ be the set of parties corrupted by $\mathcal{A}$ and $\mathbb{H}$ the set of honest parties. We assume that $\mathcal{A}$ is deterministic, and its output consists of his view during the protocol (i.e. the auxiliary input, the input of all corrupted parties, and all the messages it sends and receives). For every honest party P_j, we define a receiver-specific adversary $\mathcal{A}_j$ for the protocol Π_{bc}, and we describe it in Figure 4.2.

Figure 4.2: The Receiver Specific Adversary $\mathcal{A}_j$

Notation: To simplify the notation we assume that if $\mathcal{A}_j$ sends an abort message so does $\mathcal{A}$ but we will not write it explicitly in the rest of the description of $\mathcal{A}_j$.
First round:

- Upon receiving the first-broadcast-round message msg^1_h from a honest party P_h, $\mathcal{A}_j$ samples a uniformly random string $\mathbf{K}^b_{h\to i,\alpha}$ for every $i \in \mathcal{I}$, every $\alpha \in [\mathsf{L}]$ and every $b \in \{0,1\}$. $\mathcal{A}_j$ computes $\mathtt{GC}_h \leftarrow \mathtt{simGC}(1^\lambda, \phi(\mathsf{C}^1_{h,x_h}), \text{“}ckt\text{”})$. Then, $\mathcal{A}_j$ sends $(\mathtt{GC}_h, \mathsf{msg}^1_h, \{\mathbf{K}^b_{h\to i,\alpha}\}_{\alpha\in[\mathsf{L}],b\in\{0,1\}})$ to $\mathcal{A}$, on behalf of the honest party P_h, sends to each corrupted party P_i over $P2P$ channels in $\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4}$.

- $\mathcal{A}$ sends the message $(\mathtt{GC}_i, \mathsf{msg}^1_{i\to h}, \{\mathbf{K}^b_{i\to h,\alpha}\}_{\alpha\in[\mathsf{L}],b\in\{0,1\}})$ back, on behalf of every corrupted party P_i to every honest party P_h in $\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4}$. If there exist $i \in \mathcal{I}$ and $h, h' \in \mathbb{H}$ s.t $\mathsf{msg}^1_{i\to h} \neq \mathsf{msg}^1_{i\to h'}$ then $\mathcal{A}_j$ aborts, otherwise $\mathcal{A}_j$ broadcasts the message $\mathsf{msg}^1_{i\to j}$ for every corrupted party P_i in protocol Π_{bc}.

Second round:

- Upon receiving the second-broadcast-round message msg^2_h from a honest party P_h in Π_{bc}, $\mathcal{A}_j$ invokes the garbling scheme simulator to obtain $\{\mathbf{K}_{h,\alpha}\}_{\alpha\in[\mathsf{L}]} \leftarrow \mathtt{simGC}(1^\lambda, \mathsf{msg}^2_h, \text{“}input\text{”})$.

- Concatenate all received messages $\{\mathsf{msg}^1_{l\to h}\}_{l\in[n]}$ as: $(\mu^1_{h,1}, \ldots, \mu^1_{h,\mathsf{L}}) \leftarrow (\mathsf{msg}^1_{1\to h}, \ldots, \mathsf{msg}^1_{n\to h}) \in \{0,1\}^{\mathsf{L}}$. Denote $\mathbf{K}_{h\to i,\alpha} = \mathbf{K}^{\mu^i_{h,\alpha}}_{h\to i,\alpha}$, and $\mathcal{A}_j$ samples $\mathbf{K}_{h\to g}$ for $g \notin \mathcal{I}$ conditioning on $\mathbf{K}_{h,\alpha} = \bigoplus_{k\in[n]} \mathbf{K}_{h\to k,\alpha}$.

  $\mathcal{A}_j$ send message $(\overline{\mathtt{GC}_j}, \{\mathbf{K}_{j\to h,\alpha}\}_{j\in[n],\alpha\in[\mathsf{L}]})$ as the second round $P2P$ message for P_h in $\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4}$. $\mathcal{A}$ sends back message $(\overline{\mathtt{GC}_i}, \{\mathbf{K}_{j\to i,\alpha}\}_{j\in[n],\alpha\in[\mathsf{L}]})$, where $\overline{\mathtt{GC}_i}$ is the echo of the garbled circuits obtained in the first round.

  $\mathcal{A}_j$ checks if the set of garbled circuits echoed in round 2 by $\mathcal{A}$ in $\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4}$ are consistent with the garbled circuits received in round 1 in $\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4}$. If this is not the case, abort.

- Based on the first round messages from P_i, denote $\mathbf{K}_{i\to h,\alpha} = \mathbf{K}^{\mu^1_{h,\alpha}}_{i\to h,\alpha}$, reconstruct

$\mathbf{K}_{i,\alpha} \leftarrow \bigoplus_{k\in[n]} \mathbf{K}_{i\to k,\alpha}$, and compute $\mathsf{msg}^2_{i\to h} \leftarrow \mathtt{eval}(\mathtt{GC}_i, \{\mathbf{K}_{i,\alpha}\}_{\alpha\in[\mathsf{L}]})$. Then $\mathcal{A}_j$ broadcasts the message $\mathsf{msg}^2_{i\to j}$ for every corrupted party P_i in protocol Π_{bc}.

Third round: After receiving the third-broadcast-round message msg^3_h from a honest party P_h, $\mathcal{A}_j$ do steps similar to the first round, and broadcasts the obtained message $\mathsf{msg}^3_{i\to j}$.

Fourth round: After receiving the fourth-broadcast-round message msg^4_h, $\mathcal{A}_j$ do steps similar to the second round and broadcast the obtained message $\mathsf{msg}^4_{i\to j}$, outputs whatever $\mathcal{A}$ outputs and halts.

By the security of Π_{bc}, for every $j \in \mathbb{H}$, there exists a PPT simulator $\mathcal{S}_j$ for the adversarial strategy $\mathcal{A}_j$, such that for every auxiliary input aux, for all $x \in (\{0,1\}^*)^n$, for all $\lambda \in \mathbb{N}$, it holds that

$$\mathtt{REAL}_{\Pi_{\mathsf{bc}},\mathcal{I},\mathcal{A}_j(\mathsf{aux})}(x, 1^\lambda) \stackrel{c}{\equiv} \mathtt{IDEAL}^{\mathsf{un\text{-}abort}}_{f,\mathcal{I},\mathcal{S}_j(\mathsf{aux})}(x, 1^\lambda).$$

Every simulator $\mathcal{S}_j$ will start by extracting corrupted parties' inputs $\mathbf{x}'_j = \{x'_{i,j}\}_{i\in\mathcal{I}}$, and send them to the ideal functionality. Upon receiving the output value y, $\mathcal{S}_j$ can send a message $\mathtt{abort}$ or not send it to the ideal functionality. Finally, the outputs the simulated view (in Π_{bc}) of the adversary as follows: $\mathsf{view}_j = (\hat{\mathsf{aux}}^j, \{\hat{x}^j_i\}_{i\in\mathcal{I}}, \{\hat{\mathsf{msg}}^{l,j}_k\}_{k\in[n],l\in\{1,2,3,4\}})$

We have the following simulator $\mathcal{S}$ (in Figure 4.3) that use $\mathcal{S}' = \mathcal{S}_j$, where j is the minimal index s.t. $j \in \mathbb{H}$, and $\mathcal{A}_j$ is the corresponding receiver specific adversary. Note that $\mathcal{S}$ runs expected polynomial time because of the following reasons: (1) $\mathcal{S}_j$ is expected polynomial time, (2) all the other computations made by $\mathcal{S}$ are polynomial time.

Figure 4.3: The Simulator $\mathcal{S}$

The simulator $\mathcal{S}$ starts by invoking $\mathcal{S}'$ and let her interact with $\mathcal{A}_j$ using (sufficiently long randomness ρ, in particular for running $\mathcal{A}$ -inside $\mathcal{A}_j$- is used randomness $\rho_{\mathcal{A}}$) to invoke $\mathcal{S}'$ and $\mathcal{A}_j$. During this interaction, $\mathcal{S}'$ could invoke the ideal functionality w.r.t. adversarial input $\mathbf{x}' = \{x'_i\}_{i\in\mathcal{I}}$, in this case, $\mathcal{S}$ will forward received messages to the ideal functionality receiving the output y which is forwarded to $\mathcal{S}'$. Moreover, $\mathcal{S}'$ could abort specifying a set of indices $\mathcal{J}$ and so it does $\mathcal{S}$ sending an abort message with indices $\mathcal{J}$ to the ideal functionality. In the end of the interaction $\mathcal{S}'$ produces the following simulated view: $\mathsf{view}' = (\hat{\mathsf{aux}}, \{\hat{x}_i\}_{i\in\mathcal{I}}, \{\hat{\mathsf{msg}}^l_k\}_{k\in[n],l\in\{1,2,3,4\}})$. Then, the simulator $\mathcal{S}$ starts $\mathcal{A}$ using randomness $\rho_{\mathcal{A}}$ and executes the following steps:

First round:

- $\mathcal{S}$ samples a uniformly random string $\mathbf{K}^b_{h\to i,\alpha}$ for every $i \in \mathcal{I}$, every $\alpha \in [\mathsf{L}]$ and $b \in \{0,1\}$. $\mathcal{S}$ computes $\mathtt{GC}_h \leftarrow \mathtt{simGC}(1^\lambda, \phi(\mathtt{C}^1_h), \text{“}ckt\text{”})$ Then $\mathcal{S}$ send $\mathcal{A}$ the message $(\mathtt{GC}_h, \hat{\mathsf{msg}}^1_h, \{\mathbf{K}^b_{h\to i,\alpha}\}_{\alpha\in[\mathsf{L}]})$ on behalf of the honest party P_h to every corrupted P_i, and $\mathcal{A}$ sends back $(\mathtt{GC}_i, \mathsf{msg}^1_{i\to h}, \{\mathbf{K}^b_{i\to h,\alpha}\}_{\alpha\in[\mathsf{L}]})$

Second round:

- If for every $i \in \mathcal{I}$ there exists a value $\hat{\mathsf{msg}}^1_i$ s.t. $\hat{\mathsf{msg}}^1_i = \mathsf{msg}^1_{i\to h}$ for every $h \in \mathbb{H}$ (It means $\mathcal{A}$ sends the consistent messages), then
  - For every honest party P_h, compute $\{\mathbf{K}_{h,\alpha}\}_{\alpha\in[\mathsf{L}]} \leftarrow \mathtt{simGC}(1^\lambda, \hat{\mathsf{msg}}^2_h, \text{“}input\text{”})$.
  - Concatenate all messages $\{\hat{\mathsf{msg}}^1_l\}_{l\in[n]}$ as: $(\mu^1_1, \ldots, \mu^1_{\mathsf{L}}) \leftarrow (\hat{\mathsf{msg}}^1_1, \ldots, \hat{\mathsf{msg}}^1_n)$. For every $i \in \mathcal{I}$ and $\alpha \in [\mathsf{L}]$, denote $\mathbf{K}_{h\to i,\alpha} \leftarrow \mathbf{K}^{\mu^1_\alpha}_{h\to i,\alpha}$. Then sample uniformly random strings $\mathbf{K}_{h\to j,\alpha}$ for $j \notin \mathcal{I}$, conditioning on $\mathbf{K}_{h,\alpha} = \bigoplus_{k\in[n]} \mathbf{K}_{h\to k,\alpha}$.

- Let $\overline{\mathsf{GC}}_h$ the set of garble circuits received from the other parties in the first round.
- Send the message $(\overline{\mathsf{GC}_h}, \{\mathbf{K}_{j\to h,\alpha}\}_{j\in[n],\alpha\in[\mathsf{L}]})$ to $\mathcal{A}$ as the second round message in $\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4}$.

- If there exists $i \in \mathcal{I}$ and $h, h' \in \mathbb{H}$ s.t. $\mathsf{msg}^1_{i\to h} \neq \mathsf{msg}^1_{i\to h'}$ (It means $\mathcal{A}$ sends the inconsistent messages), then
  - Send $(\mathtt{abort}, \mathbb{H})$ to the ideal functionality.
  - For every honest party P_h, compute $\{\mathbf{K}_{h,\alpha}\}_{\alpha\in[\mathsf{L}]} \leftarrow \mathtt{simGC}(1^\lambda, \mathsf{C}^1_h(0^\mathsf{L}), \text{“}input\text{”})$.
  - For every $j \in \mathbb{H}$, let $\mathsf{msg}^1_{h\to j} \leftarrow \hat{\mathsf{msg}}^1_h$.
  - Concatenate all message $\{\mathsf{msg}^1_{l\to h}\}_{l\in[n]}$ as: $(\mu^1_{h,1}, \ldots, \mu^1_{h,\mathsf{L}}) \leftarrow (\mathsf{msg}^1_{1\to h}, \ldots, \mathsf{msg}^1_{n\to h})$. For every $i \in \mathcal{I}$ and $\alpha \in [\mathsf{L}]$, denote $\mathbf{K}_{h\to i,\alpha} \leftarrow \mathbf{K}^{\mu^1_{h,\alpha}}_{h\to i,\alpha}$. Then sample uniformly random string $\mathbf{K}_{h\to j,\alpha}$ for $j \in \mathbb{H}$.
  - Let $\overline{\mathsf{GC}}_h$ the set of garble circuits received from the other parties in the first round.
  - Send the message $(\overline{\mathsf{GC}}_h, \{\mathbf{K}_{j\to h,\alpha}\}_{j\in[n],\alpha\in[\mathsf{L}]})$ to $\mathcal{A}$ as the second round message in $\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4}$.

- $\mathcal{A}$ sends back message $(\overline{\mathsf{GC}}_i, \{\mathbf{K}_{j\to i,\alpha}\}_{j\in[n],\alpha\in[\mathsf{L}]})$. If $\mathcal{A}$ echoed inconsistent garble circuits in the second round abort, otherwise proceeds to the next step.

  If $\mathcal{A}$ sends consistent first round messages of $\Pi_{\mathtt{bc}}$, and $\mathcal{S}'$ does not abort, based on the first round messages from P_i, denote $\mathbf{K}_{i\to h,\alpha} = \mathbf{K}^{\mu^1_\alpha}_{i\to h,\alpha}$, reconstruct $\mathbf{K}_{i,\alpha} \leftarrow \bigoplus_{k\in[n]} \mathbf{K}_{i\to k,\alpha}$, and compute $\mathsf{msg}^2_{i\to h} \leftarrow \mathtt{eval}(\mathsf{GC}_i, \{\mathbf{K}_{i,\alpha}\}_{\alpha\in[\mathsf{L}]})$. Also, check that $\mathbf{K}_{h\to i,\alpha} = \mathbf{K}^{\mu^1_\alpha}_{h\to i,\alpha}$. If evaluation or check fails, this honest party P_h is added to the set $\mathcal{J}$.

- $\mathcal{S}$ sends $(\mathtt{abort}, \mathcal{J})$ to the ideal functionality.

Third round: Use message $\hat{\mathsf{msg}}^3_h$ and do the steps similar to the first round (aborting also when honest parties receive different garble circuits w.r.t. some corrupted party P_j).
Fourth round: Perform the steps similar to the second round. $\mathcal{S}$ will output whatever $\mathcal{A}$ outputs, and halt.

We use the following hybrid experiments to prove real/ideal indistinguishability:

- $\mathtt{Expt}^0_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$: In this experiment, the simulator $\mathcal{S}_0$ have the access to the internal state of the ideal functionality. $\mathcal{S}_0$ can see the honest parties input and choose their outputs. Therefore, $\mathcal{S}_0$ emulates the honest parties in the protocol $\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4}$ based on the inputs, and also sets the output of each honest parties. $\mathtt{REAL}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$ and $\mathtt{Expt}^0_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$ are identically distributed.

- $\mathtt{Expt}^1_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$: this experiment is identical to the previous experiment but the simulator $\mathcal{S}_1$ invokes $\mathcal{S}'$ and let her interact with $\mathcal{A}_j$ using (sufficiently long randomness ρ, in particular for running $\mathcal{A}$ -inside $\mathcal{A}_j$- is used randomness $\rho_\mathcal{A}$) to invoke $\mathcal{S}'$ and $\mathcal{A}_j$. During this interaction, $\mathcal{S}'$ could invoke the ideal functionality w.r.t. adversarial input $\mathbf{x}' = \{x'_i\}_{i\in\mathcal{I}}$,

in this case, $\mathcal{S}$ will forward received messages to the ideal functionality receiving the output y which is forwarded to $\mathcal{S}'$. Moreover, $\mathcal{S}'$ could abort specifying a set of indices $\mathcal{J}$ and so it does $\mathcal{S}$ sending an abort message with indices $\mathcal{J}$ to the ideal functionality.

Then, the simulator $\mathcal{S}$ starts $\mathcal{A}$ using randomness $\rho_{\mathcal{A}}$ and emulates the honest parties in the protocol $\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4}$, based on their inputs, i.e., the simulator behaves as in the previous experiment. In particular, $\mathcal{S}_1$ performs the following checks. $\mathcal{S}_1$ checks if the adversary sends inconsistent garble circuits in the first round. In this case she sends $(\texttt{abort}, \mathbb{H})$ to the ideal functionality before the third round. Moreover, $\mathcal{S}_1$ checks whether every honest party can evaluate the garbled circuits with the garbled labels from $\mathcal{A}$ in the second round (and the fourth round). Let $\mathcal{J}$ denote the set of honest parties for which the above checks failed, then $\mathcal{S}_1$ sends $(\texttt{abort}, \mathcal{J})$ to the ideal functionality. The indistinguishability between the hybrids follows from lemma 2.

- $\texttt{Expt}^2_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$: this experiment is identical to the previous experiment but other than computing $\mathbf{K}^b_{h\to k,\alpha}$ for $b \in \{0,1\}, h \in \mathbb{H}, k \in [n], \alpha \in [\mathsf{L}]$, s.t. $\mathbf{K}^b_{h,\alpha} = \bigoplus_{k\in[n]} \mathbf{K}^b_{h\to k,\alpha}$, the simulator $\mathcal{S}_2$ sends random shares (in the first round and the third round) on behalf of every honest party P_h to every corrupted party P_i. Then:

  - If $\mathcal{A}$ sends consistent messages in the first round, it means every honest party will receive the same messages from each corrupted party, and we denoted the concatenated messages as $\{\mu^1_\alpha\}_{\alpha\in[\mathsf{L}]}$. Then denote $\mathbf{K}_{h\to i,\alpha} \leftarrow \mathbf{K}^{\mu^1_\alpha}_{h\to i,\alpha}$, and sample uniformly random string $\mathbf{K}_{h\to j,\alpha}$ for $j \notin \mathcal{I}$ conditioned on $\mathbf{K}_{h,\alpha} = \bigoplus_{k\in[n]} \mathbf{K}_{h\to k,\alpha}$. These shares are used in the second round messages. Similar steps will be done in the third round to generate the fourth round messages if consistent third round messages are received.
  - If $\mathcal{A}$ send inconsistent messages (i.e. at least one corrupted party sends different messages to two honest parties), then denote $\{\mu^1_{h,\alpha}\}_{\alpha\in[\mathsf{L}]}$ as the concatenated messages, and denote $\mathbf{K}_{h\to i,\alpha} \leftarrow \mathbf{K}^{\mu^1_{h,\alpha}}_{h\to i,\alpha}$. Sample uniformly random string $\mathbf{K}_{h\to j,\alpha}$ for $j \in \mathbb{H}$ without conditions. These shares are part of the second round message. Similarly, if $\mathcal{A}$ sends inconsistent third-round messages, these steps are done in the fourth round to generate the fourth-round messages.

  The indistinguishability between the hybrids follows from lemma 3.

- $\texttt{Expt}^3_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$: this experiment is identical to the previous experiment but other than generating garble circuit honestly, the simulator $\mathcal{S}_3$ will use $\texttt{simGC}$ to generate garbled circuit. Specifically, in the first round is computed $\texttt{GC}_h \leftarrow \texttt{simGC}(1^\lambda, \phi(\mathsf{C}^1_{h,x_h}), \text{“}ckt\text{”})$. Then, every honest party can obtain msg^1_h by using received first round messages (and similarly msg^3_h). Then compute $\{\mathbf{K}_{h,\alpha}\}_{\alpha\in[\mathsf{L}]} \leftarrow \texttt{simGC}(1^\lambda, \mathsf{msg}^2_h, \text{“}input\text{”})$. Similar operations will also be performed in the fourth round. The indistinguishability between the hybrids follows from lemma 4.

- $\texttt{Expt}^4_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$: this experiment is identical to the previous experiment but the simulator $\mathcal{S}_4$ will use different inputs for $\texttt{simGC}$ to generate garbled circuit. Denote C^1_h as the boolean circuit for $\texttt{nxt-msg}^1_h$ with hard-wired input 0. If at least one corrupted party sends different messages to two honest parties $\mathcal{S}_4$ computes $\{\mathbf{K}_{h,\alpha}\}_{\alpha\in[\mathsf{L}]} \leftarrow \texttt{simGC}(1^\lambda, \mathsf{C}^1_{h,x_h}(0^\mathsf{L}), \text{“}input\text{”})$ otherwise she computes $\{\mathbf{K}_{h,\alpha}\}_{\alpha\in[\mathsf{L}]} \leftarrow \texttt{simGC}(1^\lambda, \mathsf{msg}^2_h, \text{“}input\text{”})$. Similar operations will also be performed in the third and the fourth round. The indistinguishability between the hybrids follows from lemma 5.

- $\mathtt{Expt}^{5}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$: this experiment is identical to the previous experiment but the simulator $\mathcal{S}_5$ will use the simulated messages from $\mathcal{S}'$. In the first round, P_h will send $\hat{\mathsf{msg}}^1_h$. In the second round, $\mathcal{S}_5$ computes $(\mathtt{GC}_h, \{\mathbf{K}_{h,\alpha}\}_{\alpha\in[\mathsf{L}]}) \leftarrow \mathtt{simGC}(1^\lambda, \mathsf{C}^1_h, \hat{\mathsf{msg}}^2_h)$. Similar operations will also be performed in the third and the fourth round. In this case, $\mathcal{S}_5$ does not need to access to the internal state of the ideal functionality, and it is exactly the same as $\mathcal{S}$, so $\mathtt{Expt}^{5}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$ is indistinguishable from $\mathtt{IDEAL}^{\mathsf{sa\text{-}abort}}_{f,\mathcal{I},\mathcal{S}}$. The indistinguishability between the hybrids follows from lemma 6.

Lemma 2. $\mathtt{Expt}^{0}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}} \stackrel{c}{=} \mathtt{Expt}^{1}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$*: This relies on the security of* $\Pi_{\mathtt{bc}}$ *and the correctness of the garbling scheme.*

- If $\mathcal{A}$ sends inconsistent first round (or third round) messages, in $\mathtt{Expt}^{0}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$, all honest parties will abort; in $\mathtt{Expt}^{1}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$, $\mathcal{S}_1$ will send $(\mathtt{abort}, \mathbb{H})$ to the ideal functionality, and achieve the same results as in $\mathtt{Expt}^{0}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$.

- If $\mathcal{A}$ sends the consistent first round (and third round) messages, $\mathcal{A}$ can reconstruct the second round (fourth round) messages in $\Pi_{\mathtt{bc}}$, and then send garbled circuits and garbled labels to honest party. If honest parties failed to evaluate the garbled circuits, the echoed garbled circuits were inconsistent, or $\mathcal{A}$ sent incorrect shares for honest parties' garbled labels, they will abort, and $\mathcal{S}_1$ sends $(\mathtt{abort}, \mathcal{J})$ to the ideal functionality, where $\mathcal{J}$ contains the set of indices for which the checks fail. Otherwise, by the correctness of the garbling scheme, the honest parties will receive the correct second round (fourth round) messages. In this case, if is possible to distinguish between $\mathtt{Expt}^{0}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$ and $\mathtt{Expt}^{1}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$, it is possible to distinguish between the output of $\mathcal{S}'$ and one of the receiver-specific adversary $\mathcal{A}_j$, which contradicts the security of $\Pi_{\mathtt{bc}}$.

- It is important to note that $\mathcal{A}$ can send inconsistent garbled circuits for instance $\mathtt{GC}_i$ and $\mathtt{GC}'_i$ which outputs two different second round messages. If this is the case we could construct a reduction that violates the security of $\Pi_{\mathtt{bc}}$, which aborts in the end of the second round since inconsistent garbled circuits are sent and this is detected by honest parties in the protocol at the beginning of the third round.

  A similar reduction can be performed if different garbed circuits (and therefore different 4th round messages) are sent in the last round. Note that this reduction crucially relies on the assumption that $\mathcal{S}_1$ extracts the input of the adversary from the first three rounds, and from the fact that $\mathcal{S}_1$ does note need the 4th round of the adversary to generate the honest party fourth round since the adversary could be rushing.

Lemma 3. $\mathtt{Expt}^{1}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}} \stackrel{p}{=} \mathtt{Expt}^{2}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$*: This relies on the perfect security of additive secret sharing.*

If $\mathcal{A}$ sends inconsistent first round messages, In $\mathtt{Expt}^{1}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$, the adversary can choose which share (i.e. $\mathbf{K}^0_{h,\alpha}$ and $\mathbf{K}^1_{h,\alpha}$ for honest party P_h) to see, however $\mathcal{A}$ can not recover the garbled label correctly because it does not receive sufficient amount of shares; If $\mathcal{A}$ send inconsistent first round messages in $\mathtt{Expt}^{2}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$, the adversary receives random shares that are irrelevant to the actual garbled labels. Due to the perfect security of additive secret sharing the hybrids are perfectly indistinguishable.

Lemma 4. $\mathtt{Expt}^{2}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}} \stackrel{s}{=} \mathtt{Expt}^{3}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$*: This relies on the security of garbling schemes.*

We need to prove it through $2n+2$ different hybrids. In hybrid 0, it is equivalent to $\mathtt{Expt}^{2}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$. In the hybrid H^j_i for $0 < i < n$, the party P_h for $h < i$ use $\mathtt{simGC}$ to obtain the garbled circuits in the j-th round (with $j = 2$ and $j = 4$), other parties generate the garbled circuits honestly. In hybrid n, it is equivalent to $\mathtt{Expt}^{3}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$. Assuming there exists an adversary $\mathcal{A}_{\mathcal{D}}$ can distinguish between hybrid h^j_i and hybrid h^j_{i+1} , then we can construct $\mathcal{A}'_{\mathcal{D}}$ that can break the security of the garbling scheme by the following reduction (the reduction consider the case with $j = 2$, the one for $j = 4$ works in a very similar therefore omitted):

- $\mathcal{A}'_{\mathcal{D}}$ sends to the challenger $\mathtt{C}^1_i$ obtaining $\mathtt{GC}_i$.
- On behalf of other honest parties, $\mathcal{A}'_{\mathcal{D}}$ runs the first round according to both h^j_i and h^j_{i+1} while for the for i-th party uses $\mathtt{GC}_i$. $\mathcal{A}'_{\mathcal{D}}$ receives first-round messages from $\mathcal{A}_{\mathcal{D}}$.
- $\mathcal{A}'_{\mathcal{D}}$ computes msg^2_h as $\mathcal{S}_2$ would do and sends it to the challenger obtaining $\{\mathbf{K}_{i,\alpha}\}_{\alpha\in[\mathsf{L}]}$.
- On behalf of other honest parties, $\mathcal{A}'_{\mathcal{D}}$ runs the second round according to both h^j_i and h^j_{i+1} while for the for i-th party uses $\{\mathbf{K}_{i,\alpha}\}_{\alpha\in[\mathsf{L}]}$.
- $\mathcal{A}'_{\mathcal{D}}$ does the third rounds and fourth rounds according to both h^j_i and h^j_{i+1}, and output what $\mathcal{A}_{\mathcal{D}}$ outputs.

We now observe that if $\mathcal{C}$ provide the garbled circuit and the garbled labels from real execution, then we are in hybrid h^j_i, otherwise, we are in hybrid h^j_{i+1} [14]. Because it works for all hybrids, it implies $\mathtt{Expt}^{2}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}} \stackrel{c}{=} \mathtt{Expt}^{3}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$.

Lemma 5. $\mathtt{Expt}^{3}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}} \stackrel{c}{=} \mathtt{Expt}^{4}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$*: This relies on the security of garbling schemes.*

This reduction is similar to the one above, where we need also $2n+2$ hybrids, but every hybrids only change the garbled circuits' output. Assuming there exists an adversary $\mathcal{A}_{\mathcal{D}}$ can distinguish between 2 hybrids h^j_i and h^j_{i+1}, then $\mathcal{A}'_{\mathcal{D}}$ can break the security of the garbling scheme by the following reduction:

- $\mathcal{A}'_{\mathcal{D}}$ sends to the challenger $\mathtt{C}^1_i$ obtaining $\mathtt{GC}_i$.
- On behalf of other honest parties, $\mathcal{A}'_{\mathcal{D}}$ runs the first round according to both h^j_i and h^j_{i+1} while for the for i-th party uses $\mathtt{GC}_i$. $\mathcal{A}'_{\mathcal{D}}$ receives first-round messages from $\mathcal{A}_{\mathcal{D}}$.
- $\mathcal{A}'_{\mathcal{D}}$ computes msg^2_h as $\mathcal{S}_2$ would do and sends $(\mathsf{msg}^2_h, 0^{\mathsf{L}})$ to the challenger obtaining $\{\mathbf{K}_{i,\alpha}\}_{\alpha\in[\mathsf{L}]}$.
- On behalf of other honest parties, $\mathcal{A}'_{\mathcal{D}}$ runs the second round according to both h^j_i and h^j_{i+1} while for the for i-th party uses $\{\mathbf{K}_{i,\alpha}\}_{\alpha\in[\mathsf{L}]}$.
- $\mathcal{A}'_{\mathcal{D}}$ does the third rounds and fourth rounds according to both h^j_i and h^j_{i+1}, and output what $\mathcal{A}_{\mathcal{D}}$ outputs.

[14] The reduction to run in strictly polynomial time cut the running time of the simulator $\mathcal{S}_2$ similar to what describe in Theorem 16.

We now observe that if $\mathcal{C}$ provide the garbled circuit and the garbled labels from $\mathtt{simGC}$ with circuit $\mathtt{C}^1_{h,x_h}$, then we are in hybrid h^j, otherwise we are in hybrid h^j_{i+1}[14]. Because it works for all hybrids, it implies $\mathtt{Expt}^3_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}} \stackrel{c}{=} \mathtt{Expt}^4_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$.

Lemma 6. $\mathtt{Expt}^4_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}} \stackrel{c}{=} \mathtt{Expt}^5_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$*: This relies on the security of* $\Pi_{\mathtt{bc}}$.

Note that only when the adversary sends consistent messages we are generating the corresponding next honest message invoking the underlying simulator $\mathcal{S}_j$ of $\Pi_{\mathtt{bc}}$. Moreover, if $\mathcal{A}$ sends inconsistent messages in the first 3 rounds $\mathcal{A}_j$ aborts. In this way, we ensure that the messages used by $\mathcal{S}'$ are the one from which $\mathcal{S}_j$ extracts the corrupted parties' inputs in the first 3 rounds. therefore, if there exists an adversary that can distinguish between $\mathtt{Expt}^4_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$ and $\mathtt{Expt}^5_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$, it can be used to distinguish the output between $\mathcal{S}'$ and $\mathcal{A}_j$, which violates the security of $\Pi_{\mathtt{bc}}$.

By the above demonstrations, we know $\mathtt{Expt}^0_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}} \stackrel{c}{=} \mathtt{Expt}^5_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$, therefore we can conclude that $\mathtt{REAL}_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}} \stackrel{c}{=} \mathtt{IDEAL}^{\mathsf{sa\text{-}abort}}_{f,\mathcal{I},\mathcal{S}}$.

□

4.2 P2P³-BC, UA, Plain Model, n > t

The protocol described in Figure 4.1 achieves unanimous abort security (against the same corruption threshold) when the last round is executed over the broadcast channel.

The security follows intuitively from the fact that in this case, the honest parties rely on the last round (over broadcast) to recover the output unanimously. In more detail, if any inconsistency is detected in any round before the last round, the honest party aborts signaling to abort to everybody else. Instead, if the last round is executed then the additive shares corresponding to the fourth-round next-message garbled circuits are being broadcast (instead of being sent over peer-to-peer channels), and the adversary can no longer enable only a strict subset of honest parties to evaluate the garbled circuits successfully and obtain the output. Lastly, we point that unlike the case of $P2P^4$, SA protocol in Figure 4.1, we need not assume that $\Pi_{\mathtt{bc}}$ is such that its simulator can extract inputs before the last round. This is because in this case, the last round of the UA protocol is over broadcast. Therefore any attack in the last round of this protocol directly translates to an attack in the last round of $\Pi_{\mathtt{bc}}$.

More formally, we have the following theorem.

Theorem 7 (*P2P-P2P-P2P-BC*, UA, Plain Model, $n > t$)**.** *Let f be an efficiently computable n-party function, where $n > t$. Let $\Pi_{\mathtt{bc}}$ be a BC-BC-BC-BC protocol that securely computes f with unanimous abort security against $t < n$ corruptions. Then, assuming secure garbling schemes (Definition 2), the protocol from Figure 4.1 can compute f with unanimous-abort security by a four-round protocol, where the broadcast channel is used only in the last round (while the first three rounds use peer-to-peer channels).*

Proof. The proof proceeds similarly to the one described in Theorem 1, detail follows.

The description of the receiver-specific adversary is the same. Also, the description of the simulator is the same, but in case of an abort, the simulator does not specify any index but instructs the ideal functionality to abort for all the honest parties.

The hybrids described in the proof of Theorem 1 are the same (but also the hybrid in case of an abort , instructs the ideal functionality to abort for all the honest parties).

The indistinguishability between $\texttt{Expt}^0_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$ and $\texttt{Expt}^1_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$ follows similar arguments to the proof of indistinguishability in Theorem 1, but in this case, we need to argue that the honest parties unanimously abort or recover the output. This follows from the subsequent observations:

- If $\mathcal{A}$ sends inconsistent first round (or third round) messages, in $\texttt{Expt}^0_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$, all honest parties abort; in $\texttt{Expt}^1_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$, and the simulator of $\texttt{Expt}^1_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$ (called $\mathcal{S}_1$) sends $\texttt{abort}$ to the ideal functionality, and achieve the same results as in $\texttt{Expt}^0_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$.

- If $\mathcal{A}$ sends the consistent first round or (third round) messages, $\mathcal{A}$ can reconstruct the second round (fourth round) messages in $\Pi_{\mathtt{bc}}$, and then send garbled circuits and garbled labels to honest party. If in the second round, an honest party failed to evaluate the labels, or $\mathcal{A}$ sends incorrect shares for honest parties' garbled labels, then the honest party aborts signaling to abort to everybody else. If one of the above-described failures happens in the 4th round then all the parties unanimously abort since they received the same set of shares for reconstructing the labels (due to the fact that the 4th round is over broadcast). Therefore, in both cases the simulator can instruct the ideal functionality to abort for all the honest parties. Otherwise, by the correctness of the garbling scheme, the honest parties will receive the correct second-round (fourth-round) messages. In this case, if is possible to distinguish between $\texttt{Expt}^0_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$ and $\texttt{Expt}^1_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$, it is possible to show a reduction against the security of $\Pi_{\mathtt{bc}}$.

- It is important to note that $\mathcal{A}$ can send inconsistent garbled circuits for instance $\mathtt{GC}_i$ and $\mathtt{GC}'_i$ which outputs two different second round messages. If this is the case we could construct a reduction that violates the security of $\Pi_{\mathtt{bc}}$, which aborts in the end of the second round since inconsistent garbled circuits are sent.

  The 4th round is over broadcast therefore all the honest parties abort or recover the output unanimously, since all the parties receive the same version of the garbled circuits. In this case, if it is possible to distinguish between $\texttt{Expt}^0_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$ and $\texttt{Expt}^1_{\Pi^{\mathsf{SA}}_{\mathsf{p2p}^4},\mathcal{I},\mathcal{A}}$, it is possible to distinguish between the output of $\mathcal{S}'$ and one of the receiver-specific adversary $\mathcal{A}_j$ breaking the security of $\Pi_{\mathtt{bc}}$.

The indistinguishability between the other hybrids follows the proof of Theorem 1. □

4.3 BC^3-P2P, SIA, Plain Model, n > t

Let us consider a protocol $\Pi_{\mathtt{bc}}$ which is a 4-round (where the broadcast channel is available in each round) IA MPC protocol secure against a dishonest majority. Moreover, let us assume that there exists a simulator for $\Pi_{\mathtt{bc}}$ which extracts the inputs of the adversary from the first three rounds. For instance, one can instantiate $\Pi_{\mathtt{bc}}$ using the protocol of [CRSW22][15].

Starting from $\Pi_{\mathtt{bc}}$ we can construct a SIA protocol Π in the same setting, where Π is defined exactly as $\Pi_{\mathtt{bc}}$ but where the last round is executed over the peer-to-peer channel. Intuitively, Π achieves SIA security since by our assumptions on $\Pi_{\mathtt{bc}}$ the simulator extracts the inputs of

[15]The protocol of [CRSW22] lifts an UA protocol to achieve IA security (where the simulator of the IA protocol uses the simulator of the UA protocol). If we consider, for instance, the simulator of the UA protocol constructed in [CCG$^+$20], this simulator extracts the inputs of the adversary from the first 3 rounds (see page 42 of [CCG$^+$19]). Therefore, for instance, by instantiating [CRSW22] with [CCG$^+$20] we obtain $\Pi_{\mathtt{bc}}$ with the desired property.

the adversary in the first three rounds, and therefore the adversarial inputs are fixed before the last round. Indeed, in the last round, the adversary can only decide if an honest party gets the output or learns the identity of cheaters (depending on the version of the last round message the adversary sends privately), but two honest parties can not obtain a different output (which is non-$\perp$). It can happen that different honest parties identify different cheaters and others recover (the same) outputs, but this is sufficient for SIA security. Finally, we note that a similar result was shown by [DRSY23], but only for the two rounds setting. We prove the following theorem.

Theorem 8 (*BC-BC-BC-P2P*, SIA, Plain Model, $n > t$)**.** *Let f be an efficiently computable n-party function, where $n > t$. Let Π_{bc} be a BC-BC-BC-BC protocol that securely computes f with identifiable abort security against $t < n$ corruptions with the additional constraint that a simulator can extract inputs before the last round. Then, f can be computed with selective identifiable-abort security by a four-round protocol, where the first three-rounds use broadcast channels and the last round uses peer-to-peer channels.*

Proof (Sketch). This proof proceeds similarly to the one shown in [DRSY23], we recall it for correctness making the appropriate modifications.

Consider a protocol Π (with an associate simulator $\mathcal{S}$) which is described as Π_{bc}, with the only difference being that the last round is sent over peer-to-peer channels. Let $\mathcal{I}$ be the set of dishonest parties and $\mathbb{H}$ be the set of honest parties, moreover let $\mathcal{S}_{\mathsf{bc}}$ be the simulator for Π_{bc}.

The simulator $\mathcal{S}$ is described as follows.

For the first three rounds $\mathcal{S}$ runs internally $\mathcal{S}_{\mathsf{bc}}$ emulating for her an adversary $\mathcal{A}_{\mathsf{bc}}$ of Π_{bc} (and the ideal functionality) while interacting as the honest parties w.r.t. the adversary $\mathcal{A}$ in the execution of Π (in particular if $\mathcal{S}_{\mathsf{bc}}$ wants to rewind the adversary, then $\mathcal{S}$ rewinds $\mathcal{A}$). At the end of the third round, by our assumptions, the simulator $\mathcal{S}_{\mathsf{bc}}$ has extracted the inputs of the corrupted parties (or sends an abort identifying some corrupted parties) and queries the ideal functionality. $\mathcal{S}$ queries the (real) ideal functionality forwarding the message of $\mathcal{S}_{\mathsf{bc}}$, obtaining the output y in case an abort did not occur. If an abort did not occur the simulation continues as follows. $\mathcal{S}$ sends y to $\mathcal{S}_{\mathsf{bc}}$ in order to obtain the last round for the honest parties which she sends to $\mathcal{A}$ in Π.

$\mathcal{S}$ needs now to instruct the honest parties if they recover the output or not, note that since the last round is over the peer-to-peer channel the adversary on behalf of the corrupted party P_j, for all $j \in \mathcal{I}$, can send different messages to different honest parties. Let $\vec{\mathsf{msg}}^4_h$ be the set of messages for honest party P_h that $\mathcal{A}$ sent in the last round.

$\mathcal{S}$ has the following strategies, for each honest party P_h with $h \in \mathbb{H}$:

1. If the adversary does not send a message to honest party P_h on behalf of any corrupted party P_j then $\mathcal{S}$ queries the ideal functionality with $\mathtt{abort}_j$ for party P_h.

2. Otherwise, $\mathcal{S}$ pretending to be the adversary $\mathcal{A}_{\mathsf{bc}}$ of Π_{bc} sends messages $\vec{\mathsf{msg}}^4_h$ to $\mathcal{S}_{\mathsf{bc}}$ in the broadcast round. If $\mathcal{S}_{\mathsf{bc}}$ sends $\mathtt{abort}_j$ for some $j \in \mathcal{I}$, then $\mathcal{S}$ queries the ideal functionality with $\mathtt{abort}_j$ for party P_h. Otherwise, $\mathcal{S}$ instructs the ideal function to let P_h recover the output. At this point, $\mathcal{S}$ rewinds $\mathcal{S}_{\mathsf{bc}}$ at the end of the 4th round (i.e., just before $\mathcal{S}$ received $\vec{\mathsf{msg}}^4_h$) and repeats the same steps (1),(2) for the other honest parties P'_h, with $h' > h$ and $h' \in \mathbb{H}$.

The indistinguishability between the real game and the simulated game follows from the security of Π_{bc} and since $\mathcal{S}_{\mathsf{bc}}$ is expected polynomial time so is $\mathcal{S}$.

□

5 Negative Results

5.1 BC^3-P2P, UA, Plain Model, n > t

At a high-level, we show that any BC^3-$P2P$ protocol achieving UA against dishonest majority implies a three-round oblivious transfer (OT) protocol in the plain model, which is known to be impossible [HV16].

Theorem 9 (*BC-BC-BC-P2P*, UA, $n > t$)**.** *There exists function f such that no n-party four-round protocol can compute f with unanimous-abort security against $t < n$ corruptions, such that the first three rounds use broadcast and point-to-point channels and the last round uses only point-to-point channels.*

Proof. For our proof, we use the function f_{mot} defined below. Let P_1 takes as input a bit $b \in \{0,1\}$. Parties P_i $(2 \leq i \leq n-1)$ hold as input $\perp$, and party P_n takes as input a pair of λ-bit strings, denoted as $X_n = (x_0, x_1)$. The function f_{mot} outputs x_b to parties P_1 and P_2 and $\perp$ to other parties. (i.e. parties P_i $(3 \leq i \leq n)$ do not learn an output). Essentially, $f_{mot}(b, \perp, \perp, \ldots, \perp, (x_0, x_1)) \rightarrow (x_b, x_b, \perp, \ldots, \perp)$ denotes the function.

Towards a contradiction, assume there exists a four-round protocol Π that does not use a broadcast channel in the last round and computes f_{mot} with unanimous abort security. The main idea of the proof is that Π can be transformed into a three-round two-party protocol Π^{ot} realizing the oblivious transfer (OT) functionality in the plain model (which is known to be impossible) [16].

Let $\mu = 1 - negl(\lambda)$ denote the overwhelming probability with which security of Π holds, where the probability is defined over the random coins used by the parties. Consider an execution where everyone behaves honestly. Let bc_j^i and $\mathsf{msg}_{j\rightarrow k}^i$ denote P_j's broadcast message and the point-to-point message sent by P_j to P_k in the ith round, in such an execution. Next, we present the following useful lemma.

Lemma 10. *Protocol Π must be such that the combined view of parties $\{P_1, P_2, \ldots, P_{\frac{n}{2}}\}$ at the end of Round 3 suffices to compute the output of f_{mot} with overwhelming probability.*

Proof. Consider a scenario in an execution of the protocol, where the adversary corrupts parties $\{P_{\frac{n}{2}+1}, \ldots, P_n\}$ (for simplicity, assume n to be an even number) and does the following.

Rounds 1 to 3: On behalf of the corrupt parties, she behaves honestly until and including Round 3. In more detail, the adversary sends messages bc_j^i and $\{\mathsf{msg}_{j\rightarrow k}^i\}_{k\in[n]}$ as per an honest execution on behalf of each corrupt P_j for Rounds $i = 1$ to 3.
Round 4: In the last round, the adversary sends point-to-point messages only to party P_2, and is silent otherwise. In more detail, the adversary sends $\mathsf{msg}_{j\rightarrow 2}^4$ on behalf of each corrupt P_j but does not send any other messages.

First, we claim that P_2 must output the correct output x_b in the above scenario with overwhelming probability p. This follows directly from the correctness of Π (which holds with probability μ) because P_2's view in the above scenario is identical to her view in an execution where everyone behaves honestly. Based on our assumption that Π achieves unanimous abort security (with probability at least μ), it must be the case that honest P_1 also outputs x_b with overwhelming probability $\mu \times \mu = \mu^2$.

[16] We point that one of the impossibility results in [PRS20] also reduces to the impossibility of three-round OT protocol in the plain model, but the starting point of their transformation is a best-of-both-worlds protocol that achieves fairness in honest majority and unanimous abort in dishonest majority. This makes their transformation different than ours as their transformation exploits the property of fairness.

Note that the only messages P_1 receives in the last round are from parties $\{P_2, \ldots, P_{\frac{n}{2}}\}$ (as she receives no communication from the other parties $\{P_{\frac{n}{2}+1}, \ldots, P_n\}$ in the last round). The fourth round point-to-point messages of honest parties $\{P_2, \ldots, P_{\frac{n}{2}}\}$ to P_1 must be computed based on their view at the end of Round 3. Therefore, we can conclude that the combined view of parties $\{P_1, P_2, \ldots, P_{\frac{n}{2}}\}$ at the end of Round 3 subsumes (i.e. contains at least as much information as) the view of an honest P_1 at the end of the execution, in the above scenario.

Since P_1 obtained the output x_b with overwhelming probability μ^2 in the above scenario, we can conclude that the combined view of parties $\{P_1, P_2, \ldots, P_{\frac{n}{2}}\}$ at the end of Round 3 must suffice to compute the output of f_{mot} with overwhelming probability as well. □

Transformation to a three-round two-party OT protocol. Next, we show how protocol Π can be used to design a three-round two-party OT protocol Π^{ot} between a sender P_S (with a pair of messages (m_0, m_1) as the input) and receiver P_R (having a choice bit c as input). The idea is to let P_S emulate the role of parties $\{P_{\frac{n}{2}+1}, \ldots, P_n\}$ using input $(x_0, x_1) = (m_0, m_1)$ on behalf of P_n (and input $\perp$ on behalf of others), while P_R emulates the role of parties $\{P_1, P_2, \ldots, P_{\frac{n}{2}}\}$ using input $b = c$ on behalf of P_1 (and input $\perp$ on behalf of others). In more detail, in the i-th round ($i \in [3]$), P_S sends messages $\{\mathsf{bc}^i_j, \{\mathsf{msg}^i_{j\to k}\}_{k\in\{1,\ldots,\frac{n}{2}\}}\}_{j\in\{\frac{n}{2}+1,\ldots,n\}}$ to P_R. Similarly, P_R sends messages $\{\mathsf{bc}^i_j, \{\mathsf{msg}^i_{j\to k}\}_{k\in\{\frac{n}{2}+1,\ldots,n\}}\}_{j\in\{1,\ldots,\frac{n}{2}\}}$ to P_S in the ith round. Next, we argue that Π^{ot} is an OT protocol (realizing the OT functionality [17]).

Lemma 11. *Π^{ot} is a secure two-party three-round protocol that realizes the oblivious transfer (OT) ideal functionality.*

Proof. First, we argue that the three-round protocol Π^{ot} is correct. This holds if the receiver P_R is able to compute the correct output i.e. m_c with overwhelming probability at the end of Π^{ot}. This is indeed the case, as P_R's view at the end of Π^{ot} corresponds to the combined view of parties $\{P_1, P_2, \ldots, P_{\frac{n}{2}}\}$ at the end of Round 3 of Π, which suffices to compute the correct output $x_b = m_c$ with overwhelming probability (Lemma 10).

Next, we argue security of Π^{ot} . Consider the case where P_R is corrupt. Since we assume that Π is secure against any adversary corrupting a subset of parties (with size at most $n-1$), it must hold that it is secure against an adversary corrupting $\{P_1, P_2, \ldots, P_{\frac{n}{2}}\}$. More specifically, such an adversary's view does not reveal any information about honest P_n's input, beyond the output x_b. Since P_R in Π^{ot} emulates the role of $\{P_1, P_2, \ldots, P_{\frac{n}{2}}\}$ in Π, we can infer that corrupt P_R in Π^{ot} does not learn any information beyond the output $x_b = m_c$ (specifically, P_R does not learn P_n's input $m_{1\oplus c}$). More formally, we can construct a simulator for Π^{ot} corresponding to the case of corrupt P_R by simply running the steps of simulator of Π (until Round 3) for the case when the adversary corrupts $\{P_1, P_2, \ldots, P_{\frac{n}{2}}\}$.

Similarly, the case when P_S is corrupt in Π^{ot} translates to the case when the adversary corrupts $\{P_{\frac{n}{2}+1}, \ldots, P_n\}$ in Π. Since Π is assumed to be secure against such an adversary, it holds that such an adversary learns nothing about the input bit b of P_1. This lets us infer that corrupt P_S in Π^{ot} learns nothing about the choice bit $b = c$ of P_R. This is because we can construct a simulator for Π^{ot} corresponding to the case of corrupt P_S by simply running the steps of the simulator of Π (until Round 3) for the case when the adversary corrupts $\{P_{\frac{n}{2}+1}, \ldots, P_n\}$. This completes the proof that Π^{ot} is a secure three-round protocol that realizes the OT functionality. □

[17] 1-out-of-2 Oblivious Transfer (OT) is a two-party functionality involving a sender with input (m_0, m_1) and a receiver with input $c \in \{0, 1\}$. Informally, the security is that the receiver should learn m_c (and nothing else) and the sender should learn nothing.

Note that Π^{ot} involves both P_S and P_R sending messages in each round of the three-round protocol, therefore the OT protocol obtained in the above transformation is bidirectional. Such a bidirectional OT protocol can be further transformed to an alternating-message OT protocol $\tilde{\Pi}^{ot}$ in a round-preserving manner [CCG+20], where each round comprises of a message from either P_S or P_R (but not both). However, such a protocol $\tilde{\Pi}^{ot}$ cannot exist due to the known impossibility of three-round alternating-message OT in the plain model [HV16] with black-box simulation; which is the final contradiction.

□

5.2 SIA Impossibility Results

Theorem 12 (*BC-BC-P2P-P2P*, SIA, Plain Model, $n > t$)**.** *Assume the existence of pseudorandom functions. There exists function f such that no n-party four-round protocol (in the plain model) can compute with selective identifiable-abort security, against $t < n$ corruptions, while in the protocol, the first two rounds use broadcast channels and the last two rounds use peer-to-peer channels.*

Proof. We start the proof assuming that the four-round protocol Π is run by three parties only, and we extend the proof to the n-party case in the end. By contradiction, assume that there exists a three-party protocol Π that can compute any function f with selective identifiable-abort security where just one party P_{out} gets the output[18] and the broadcast channel is accessible only in the first two rounds. Let us denote the three parties running the protocol Π with P_1, P_2, and P_{out}.

Consider the following adversarial strategy of Figure 5.1. In summary, in this scenario, corrupted P_1 behaves like an honest party, with the difference that it does not send the third and the fourth message to P_2, and it pretends that it does not receive the third message and the fourth message from P_2.

Figure 5.1: Scenario 1

Setting: P_1 is corrupted party P_2 and P_{out} are honest parties.
Private input: Every party P_i has a private input $x_i \in \{0,1\}^*$.

First round (BC):
Every party P_i samples the randomness r_i from uniform distribution $\mathcal{D}$, computes $\mathsf{msg}_i^1 \leftarrow \mathtt{frst\text{-}msg}_i(x_i; r_i)$, and sends the message over the broadcast channel.
Second round (BC):
Every party P_i computes $\mathsf{msg}_i^2 \leftarrow \mathtt{nxt\text{-}msg}_i^1(x_i, \{\mathsf{msg}_j^1\}_{j\in\{1,2,\mathsf{out}\}}; r_i)$, and sends it over the broadcast channel.
Third round (P2P):

1. Every party P_i computes $(\{\mathsf{msg}_{i\to j}^3\}_{j\in\{1,2,\mathsf{out}\}}) \leftarrow \mathtt{nxt\text{-}msg}_i^2(x_i, \{\mathsf{msg}_j^k\}_{j\in\{1,2,\mathsf{out}\},k\in\{1,2\}}; r_i)$.
2. P_1 sends $\mathsf{msg}_{1\to\mathsf{out}}^3$ to P_{out}. P_{out} sends $\mathsf{msg}_{\mathsf{out}\to 1}^3$ to P_1, and sends $\mathsf{msg}_{\mathsf{out}\to 2}^3$ to P_2. P_2 sends $\mathsf{msg}_{2\to 1}^3$ to P_1, and sends $\mathsf{msg}_{2\to\mathsf{out}}^3$ to P_{out}.

Fourth round (P2P):

1. P_1 sets $\mathsf{msg}_{2\to 1}^3 = \bot$ and computes $(\{\mathsf{msg}_{1\to j}^4\}_{j\in\{1,2,\mathsf{out}\}}) \leftarrow \mathtt{nxt\text{-}msg}_1^3(x_1, \{\mathsf{msg}_j^k\}_{j\in\{1,2,\mathsf{out}\},k\in\{1,2\}}, \{\mathsf{msg}_{j\to 1}^3\}_{j\in\{1,2,\mathsf{out}\}}; r_1)$.

[18] We are assuming implicitly this requirement on f thought the rest of the proof.

2. P_{out} computes $(\{\mathsf{msg}^4_{2\to j}\}_{j\in\{1,2,\mathsf{out}\}}) \leftarrow \mathtt{nxt\text{-}msg}^3_{\mathsf{out}}(x_{\mathsf{out}}, \{\mathsf{msg}^k_j\}_{j\in\{1,2,\mathsf{out}\},k\in\{1,2\}}, \{\mathsf{msg}^3_{j\to\mathsf{out}}\}_{j\in\{1,2,\mathsf{out}\}}; r_{\mathsf{out}})$.

3. P_2 computes $(\{\mathsf{msg}^4_{2\to j}\}_{j\in\{1,2,\mathsf{out}\}}) \leftarrow \mathtt{nxt\text{-}msg}^3_2(x_2, \{\mathsf{msg}^k_j\}_{j\in\{1,2,\mathsf{out}\},k\in\{1,2\}}, \{\mathsf{msg}^3_{j\to 2}\}_{j\in\{2,\mathsf{out}\}}; r_2)$.

4. P_1 sends $\mathsf{msg}^4_{1\to\mathsf{out}}$ to P_{out}. P_{out} sends $\mathsf{msg}^4_{\mathsf{out}\to 1}$ to P_1, and sends $\mathsf{msg}^4_{\mathsf{out}\to 2}$ to P_2. P_2 sends $\mathsf{msg}^4_{2\to\mathsf{out}}$ to P_{out} and $\mathsf{msg}^4_{2\to 1}$ to P_1.

Given the above adversarial strategy, we proceed now in a series of steps in order to reach a contradiction.

Step 1: $\mathbf{P}_{\mathsf{out}}$ can not abort identifying the corrupted party. We prove that, if P_{out} aborts, it can not identify that P_1 aborted. We prove this by contradiction. Consider the scenario of Figure 5.2. In this, the corrupted P_2 behaves like an honest party, and he does not send the third and the fourth round message to P_1. At the same time, it pretends that it does not receive the third round and fourth message from P_1. P_1 behaves honestly, sending all the messages that the protocol Π prescribes. In summary, P_2 behaves like P_1 behaves in Scenario 1.

Figure 5.2: Scenario 2

Setting: P_2 is corrupted party. P_1 and P_{out} are honest parties.
Private input: Every party P_i has a private input $x_i \in \{0,1\}^*$.

First round (BC):
Every P_i samples the randomness r_i from uniform distribution $\mathcal{D}$, computes $\mathsf{msg}^1_i \leftarrow \mathtt{frst\text{-}msg}_i(x_i; r_i)$, and sends the message over the broadcast channel.
Second round (BC):
Every party P_i computes $\mathsf{msg}^2_i \leftarrow \mathtt{nxt\text{-}msg}^1_i(x_i, \{\mathsf{msg}^1_j\}_{j\in\{1,2,\mathsf{out}\}}; r_i)$, and sends it over the broadcast channel.
Third round (P2P):

1. Every party P_i computes $(\{\mathsf{msg}^3_{i\to j}\}_{j\in\{1,2,\mathsf{out}\}}) \leftarrow \mathtt{nxt\text{-}msg}^2_i(x_i, \{\mathsf{msg}^k_j\}_{j\in\{1,2,\mathsf{out}\},k\in\{1,2\}}; r_i)$.

2. P_2 sends $\mathsf{msg}^3_{2\to\mathsf{out}}$ to P_{out}. P_{out} sends $\mathsf{msg}^3_{\mathsf{out}\to 1}$ to P_1, and sends $\mathsf{msg}^3_{\mathsf{out}\to 2}$ to P_2. P_1 sends $\mathsf{msg}^3_{1\to\mathsf{out}}$ to P_{out}, and sends $\mathsf{msg}^3_{1\to 2}$ to P_2.

Fourth round (P2P):

1. P_1 computes $(\{\mathsf{msg}^4_{1\to j}\}_{j\in\{1,2,\mathsf{out}\}}) \leftarrow \mathtt{nxt\text{-}msg}^3_1(x_1, \{\mathsf{msg}^k_j\}_{j\in\{1,2,\mathsf{out}\},k\in\{1,2\}}, \{\mathsf{msg}^3_{j\to 1}\}_{j\in\{1,\mathsf{out}\}}; r_1)$.

2. P_{out} computes $(\{\mathsf{msg}^4_{\mathsf{out}\to j}\}_{j\in\{1,2,\mathsf{out}\}}) \leftarrow \mathtt{nxt\text{-}msg}^3_{\mathsf{out}}(x_{\mathsf{out}}, \{\mathsf{msg}^k_j\}_{j\in\{1,2,\mathsf{out}\},k\in\{1,2\}}, \{\mathsf{msg}^3_{j\to\mathsf{out}}\}_{j\in\{1,2,\mathsf{out}\}}; r_{\mathsf{out}})$.

3. P_2 sets $\mathsf{msg}^3_{1\to 2} = \perp$ and computes $(\{\mathsf{msg}^4_{2\to j}\}_{j\in\{1,2,\mathsf{out}\}}) \leftarrow \mathtt{nxt\text{-}msg}^3_2(x_2, \{\mathsf{msg}^k_j\}_{j\in\{1,2,\mathsf{out}\},k\in\{1,2\}}, \{\mathsf{msg}^3_{j\to 2}\}_{j\in\{1,2,\mathsf{out}\}}; r_2)$.

4. P_2 sends $\mathsf{msg}^4_{2\to\mathsf{out}}$ to P_{out}. P_{out} sends $\mathsf{msg}^4_{\mathsf{out}\to 1}$ to P_1, and sends $\mathsf{msg}^4_{\mathsf{out}\to 2}$ to P_2. P_1 sends $\mathsf{msg}^4_{1\to\mathsf{out}}$ to P_{out} and $\mathsf{msg}^4_{1\to 2}$ to P_2.

Intutively, in Scenario 1 (in Figure 5.1), P_2 can potentially report P_1's misbehaviour to P_{out} earliest in round 4 (since P_1 behaved honestly in round 1 and round 2). This means that P_{out} cannot identify the corrupted party (and abort) until all the four rounds are received. However, a corrupted P_1 is pretending that P_2 did not send the third round message. Hence, none of the messages that P_{out} receives in the fourth round would help him. In particular, P_{out} sees P_1 and P_2 blaming each other. In addition, P_{out} can not see what happened on $P2P$ channel between P_1 and P_2, therefore, P_{out} can not identify the corrupted party correctly.

Formally, if in Scenario 1 P_{out} aborts, then based on the definition of selective identifiable-abort, P_{out} identifies P_1 as the corrupted party. However, the view of P_{out} in the Scenario 1 is identical to the view of P_{out} in Scenario 2 (in Figure 5.2). Because the view of P_{out} is identical in the two scenarios, then P_{out} has the same behavior in both scenarios. Hence, P_{out} identifies P_1 as the corrupted party. However, in Scenario 2, P_1 is honest, and this contradicts the SIA security of Π. From the above, we can conclude that P_{out} does not abort in Scenario 1, hence, it must be able to compute the output.

In addition, by generalizing Step 1 above, we have the following lemma:

Lemma 13. *Let f be an efficiently computable three-party function and we denote the three party as $(P_1, P_2, P_{\mathsf{out}})$. Assume there exists a three-party protocol Π that securely computes f with selective-identifiable-abort security where the party P_{out} recovers the output, for any communication pattern $\{BC, P2P, BC\}$-BC (one of the first three rounds is using $P2P$ channels). Then when running Π against the corrupted P_1 (or P_2), whose malicious behaviors are only not sending and pretending not to receive messages from P_2 (resp. P_1) in $P2P$ rounds, P_{out} can not abort and identify the correct corrupted party, and it must obtain the output.*

Proof. The proof is similar to Step 1 of Theorem 12. More in detail, we can construct two scenarios, that in the first scenario P_1 behaves honestly in BC rounds. It does not send the message to P_2 and pretends not to receive from P_2 in the $P2P$ round. In the second scenario, P_2 behaves honestly in BC rounds. It does not send the message to P_1 and pretends not to receive from P_1 in the $P2P$ round. Then the views of P_{out} are identical in these two scenarios, and if it aborts, it can not identify the correct corrupted party, which means it must obtain the output. □

Step 2: Constructing an SA secure protocol (only for two corruption patterns). We now consider a new protocol, that we denote with Π' (and denote the parties running the protocol with P'_1, P'_2 and P'_{out}). This protocol works exactly like Π, with the following differences:

- The honest P'_1 does not send the third message to P'_2.
- No fourth messages between P'_1 and P'_2.
- P'_{out} does not send any fourth round to P'_1 and P'_2.

We prove that this protocol is secure with selective abort. Informally, this is possible because the honest parties send fewer messages compared to Π, and the party P'_{out} will still be able to compute the output due to the argument given above. Moreover, given that we just want to obtain SA security, we can remove the messages that P_{out} sends in the last round. Formally, we prove that if Π is SIA secure, then Π' (that we propose in Figure 5.3) is secure with selective abort (SA) for two corruption patterns. Namely, we prove that the protocol is secure when either P'_1 and P'_{out} are corrupted or when P'_{out} and P'_2 are corrupted. Looking ahead, we focus only on these corruption patterns, because proving the security of Π' only in these cases would be enough to reach our final contradiction. Below, we provide a more formal argument.

Figure 5.3: The new protocol Π'

Primitives: A three-party four-round protocol $\Pi = \{(\texttt{frst-msg}_i, \{\texttt{nxt-msg}_i^k\}_{k\in\{1,2,3\}}, \texttt{output}_i)\}_{i\in\{1,2,\mathsf{out}\}}$ that securely computes any f with selective identifiable-abort security against $t < n$ corruptions, where the first two rounds use the broadcast channels to exchange messages, and last two rounds use $P2P$ channels.

Private input: Every party P_i' has a private input $x_i \in \{0,1\}^*$.

First round (BC):

Every party P_i' samples the randomness r_i from uniform distribution $\mathcal{D}$, computes $\mathsf{msg}_i^1 \leftarrow \texttt{frst-msg}_i(x_i; r_i)$, and sends the message over the broadcast channel.

Second round (BC):

Every party P_i' computes $\mathsf{msg}_i^2 \leftarrow \texttt{nxt-msg}_i^1(x_i, \{\mathsf{msg}_j^1\}_{j\in\{1,2,\mathsf{out}\}}; r_i)$, and sends it over the broadcast channel.

Third round (P2P):

1. Every party P_i' computes $(\{\mathsf{msg}_{i\to j}^3\}_{j\in\{1,2,\mathsf{out}\}}) \leftarrow \texttt{nxt-msg}_i^2(x_i, \{\mathsf{msg}_j^k\}_{j\in\{1,2,\mathsf{out}\},k\in\{1,2\}}; r_i)$.
2. P_1' sends $\mathsf{msg}_{1\to\mathsf{out}}^3$ to P_{out}'. P_{out}' sends $\mathsf{msg}_{\mathsf{out}\to 1}^3$ to P_1', and sends $\mathsf{msg}_{\mathsf{out}\to 2}^3$ to P_2'. P_2' sends $\mathsf{msg}_{2\to 1}^3$ to P_1', and sends $\mathsf{msg}_{2\to\mathsf{out}}^3$ to P_{out}'.

Fourth round (P2P):

1. P_1' sets $\mathsf{msg}_{2\to 1}^3 = \perp$ and computes $(\{\mathsf{msg}_{1\to j}^4\}_{j\in\{1,2,\mathsf{out}\}}) \leftarrow \texttt{nxt-msg}_1^3(x_1, \{\mathsf{msg}_j^k\}_{j\in\{1,2,\mathsf{out}\},k\in\{1,2\}}, \{\mathsf{msg}_{j\to 1}^3\}_{j\in\{1,2,\mathsf{out}\}}; r_1)$.
2. P_{out}' computes $(\{\mathsf{msg}_{\mathsf{out}\to j}^4\}_{j\in\{1,2,\mathsf{out}\}}) \leftarrow \texttt{nxt-msg}_{\mathsf{out}}^3(x_{\mathsf{out}}, \{\mathsf{msg}_j^k\}_{j\in\{1,2,\mathsf{out}\},k\in\{1,2\}}, \{\mathsf{msg}_{j\to\mathsf{out}}^3\}_{j\in\{1,2,\mathsf{out}\}}; r_{\mathsf{out}})$.
3. P_2' computes $(\{\mathsf{msg}_{2\to j}^4\}_{j\in\{1,2,\mathsf{out}\}}) \leftarrow \texttt{nxt-msg}_2^3(x_2, \{\mathsf{msg}_j^k\}_{j\in\{1,2,\mathsf{out}\},k\in\{1,2\}}, \{\mathsf{msg}_{j\to 2}^3\}_{j\in\{2,\mathsf{out}\}}; r_2)$.
4. P_1' sends $\mathsf{msg}_{1\to\mathsf{out}}^4$ to P_{out}'. P_2' sends $\mathsf{msg}_{2\to\mathsf{out}}^4$ to P_{out}'.

Output Computation:

1. P_{out}' compute and output $y \leftarrow \texttt{output}_{\mathsf{out}}(x_{\mathsf{out}}, \{\mathsf{msg}_j^k\}_{j\in\{1,2,\mathsf{out}\},k\in\{1,2\}}, \{\mathsf{msg}_{j\to\mathsf{out}}^k\}_{j\in\{1,2,\mathsf{out}\},k\in\{3,4\}}; r_{\mathsf{out}})$

The security of the SIA protocol Π ensures us that there exist corresponding simulators for (all) the corruption patterns, we will exploit those simulators to construct the simulators for proving the security of Π'. Let $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SIA}}$ and $\mathcal{S}_{2,\mathsf{out}}^{\mathsf{SIA}}$ be the simulators of Π for, respectively, corrupted P_1 and P_{out} and for corrupted P_2 and P_{out}. We construct two new simulators $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$ and $\mathcal{S}_{2,\mathsf{out}}^{\mathsf{SA}}$ which, respectively, make use of $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SIA}}$ and $\mathcal{S}_{2,\mathsf{out}}^{\mathsf{SIA}}$, and use them to prove the security of Π' in the above-mentioned corruption patterns.

To formally do that, we need to transform an adversary $\mathcal{A}^{\mathsf{SA}}$ attacking Π' into an admissible adversary $\mathcal{M}_{\mathsf{intf}}$ of Π (we need to do that since the simulators $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SIA}}$ and $\mathcal{S}_{2,\mathsf{out}}^{\mathsf{SIA}}$ only work against adversaries attacking the protocol Π). $\mathcal{M}_{\mathsf{intf}}$ runs internally $\mathcal{A}^{\mathsf{SA}}$ and acts as a proxy for the messages between the simulator $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SIA}}$ (resp. $\mathcal{S}_{2,\mathsf{out}}^{\mathsf{SIA}}$) and $\mathcal{A}^{\mathsf{SA}}$, withholding the messages that

honest party P_1' is not supposed to send in Π'. The Figure 5.4 formally describes $\mathcal{M}_{\mathsf{intf}}$. In this, we denote as the *left interface*, the interface where the adversary sends and receives the protocol messages.

Figure 5.4: The adversary $\mathcal{M}_{\mathsf{intf}}$

Notation: Let $\mathbb{H}$ be the set of indices of the honest parties and $\mathcal{I}$ be the indices of the corrupted parties. $\mathcal{M}_{\mathsf{intf}}$ internally runs the adversary $\mathcal{A}^{\mathsf{SA}}$, and is equipped with a left interface, where it receives the messages computed on behalf of the honest parties and sends the messages computed on the behalf of the corrupted parties.

First round (BC):

1. Upon receiving msg_h^1 on the left interface with $h \in \mathbb{H}$, $\mathcal{M}_{\mathsf{intf}}$ forwards the message to $\mathcal{A}^{\mathsf{SA}}$ in Π'.
2. Upon receiving the messages sent by $\mathcal{A}^{\mathsf{SA}}$, $\mathcal{M}_{\mathsf{intf}}$ forwards them to the left interface, where it is acting as a corrupted party for Π.

Second round (BC):

1. Upon receiving msg_h^2 on the left interface, where $h \in \mathbb{H}$, $\mathcal{M}_{\mathsf{intf}}$ forwards the message to $\mathcal{A}^{\mathsf{SA}}$.
2. Upon receiving the messages sent by $\mathcal{A}^{\mathsf{SA}}$ in Π', $\mathcal{M}_{\mathsf{intf}}$ forwards them, acting as the corrupted parties in Π.

Third round (P2P):

1. Upon receiving $\mathsf{msg}_{h\to j}^3$ in the left interface, where $h \in \mathbb{H}$ and $j \in \mathcal{I}$, $\mathcal{M}_{\mathsf{intf}}$ forwards the message $\mathsf{msg}_{h\to\mathsf{out}}^3$ (and the message $\mathsf{msg}_{2\to 1}^3$ in the case where $2 \in \mathbb{H}$) to $\mathcal{A}^{\mathsf{SA}}$.
2. Upon receiving the messages sent by $\mathcal{A}^{\mathsf{SA}}$, $\mathcal{M}_{\mathsf{intf}}$ forwards them to the left interface acting as the corrupted parties in Π.

Fourth round (P2P):

1. Upon receiving $\mathsf{msg}_{h\to j}^4$ on the left interface, where $h \in \mathbb{H}$ and $j \in \mathcal{I}$, $\mathcal{M}_{\mathsf{intf}}$ forwards the message $\mathsf{msg}_{h\to\mathsf{out}}^4$ to $\mathcal{A}^{\mathsf{SA}}$ (if any).
2. Upon receiving the messages sent by $\mathcal{A}^{\mathsf{SA}}$, $\mathcal{M}_{\mathsf{intf}}$ forwards them to its left interface.

We are now ready to show how the simulator $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$ of Π' for the case where P_1' and P_{out}' are corrupted. The simulator $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$ is formally described in Figure 5.5.

Figure 5.5: $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$

$\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$ performs the following steps:

- Invoke $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SIA}}$ for the adversary $\mathcal{M}_{\mathsf{intf}}$, querying and receiving responses to and from its left interface.
- Work as a proxy between the ideal functionality and $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SIA}}$.

In the end, $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ output whatever $\mathcal{S}^{\mathsf{SIA}}_{1,\mathsf{out}}$ outputs, and halt.

For the case where P'_2 and P'_{out} are corrupted, we can define the simulator $\mathcal{S}^{\mathsf{SA}}_{2,\mathsf{out}}$ similarly to $\mathcal{S}^{\mathsf{SIA}}_{1,\mathsf{out}}$, but using $\mathcal{S}^{\mathsf{SIA}}_{2,\mathsf{out}}$. If an adversary $\mathcal{A}^{\mathsf{SA}}$ attacking Π' is able to distinguish between when it is receiving messages produced by $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ (resp. $\mathcal{S}^{\mathsf{SA}}_{2,\mathsf{out}}$) from the case when the messages are generated from an honest party running Π', then we can show an adversary that contradicts the SIA security of Π. In the reduction $\mathcal{A}^{\mathsf{SIA}}$ simply runs internally $\mathcal{M}_{\mathsf{intf}}$, which in turn it will run $\mathcal{A}^{\mathsf{SA}}$.

Step 3: Modifying adversary $\mathcal{A}^{\mathsf{SA}}$. As a stepping stone toward proving the final result, we consider first another adversary $\mathcal{A}^{\mathtt{PRF}}_{1,\mathsf{out}}$, which corrupts P'_1 and P'_{out} and acts as follows.

The corrupted parties P'_1 and P'_{out} act like the honest parties running Π' would, except that they are rushing in the first round (i.e., they wait to receive the honest party's message before sending their first round), and compute their input and randomness by evaluating a PRF on input the message received from the honest party. More formally, $\mathcal{A}^{\mathtt{PRF}}_{1,\mathsf{out}}$ samples two different keys (k_1, k_2) for a PRF $\mathtt{F}$. Upon receiving the first round msg^1_2 from the honest P'_2, the adversary computes $x_1 \leftarrow \mathtt{F}_{k_1}(\mathsf{msg}^1_2)$, $r_1 \leftarrow \mathtt{F}_{k_2}(\mathsf{msg}^1_2)$. Then $\mathcal{A}^{\mathtt{PRF}}_{1,\mathsf{out}}$ use (x_1, r_1) and original input and randomness of P'_{out} to finish all four round interactions with the honest party P'_2. We define $\mathcal{A}^{\mathtt{PRF}}_{2,\mathsf{out}}$ similarly.

We need to prove even against such an adversary there exists a simulator, that can successfully extract the input from a corrupted P'_1. A simulator for $\mathcal{A}^{\mathtt{PRF}}_{1,\mathsf{out}}$ trivially exists due to the SA security of Π', hence, we need to argue that such a simulator does query the ideal functionality, hence, it extracts the input of the corrupted parties (this will be crucial for the last step of our impossibility proof). To prove that this is indeed the case, we start by observing that, trivially, when all the parties are honest then Π' terminates and P'_{out} computes the output, with no party triggering an abort. Consider now an adversary $\mathcal{A}^{\mathsf{r}}_{1,\mathsf{out}}$, that corrupts the parties with index 1 and out, and instructs these parties to be rushing in the first round, and non-rushing in the remaining rounds, without any other change in the behaviors of corrupted parties.

Also in this case, it is easy to see that the honest party will not abort and that P'_{out} will compute the output. What remains to prove is that the view of the honest party stays the same when interacting with $\mathcal{A}^{\mathtt{PRF}}_{1,\mathsf{out}}$ instead of $\mathcal{A}^{\mathsf{r}}_{1,\mathsf{out}}$. To do that, we prove the following lemma, which holds due to the security of the PRF.

Lemma 14. *Let $\mathcal{A}^{\mathtt{PRF}}_{1,\mathsf{out}}(\mathsf{aux})$ and $\mathcal{A}^{\mathsf{r}}_{1,\mathsf{out}}$ be the adversaries described above. Assume that PRFs exist, then, for every auxiliary input aux, for all $x \in (\{0,1\}^*)^3$, for all $\lambda \in \mathbb{N}$, it holds that the probability that the honest party aborts in $\mathtt{REAL}_{\Pi,\{1,\mathsf{out}\},\mathcal{A}^{\mathtt{PRF}}_{1,\mathsf{out}}(\mathsf{aux})}(x, 1^\lambda)$ is negligible-close to the probability that the honest party aborts in $\mathtt{REAL}_{\Pi,\{1,\mathsf{out}\},\mathcal{A}^{\mathsf{r}}_{1,\mathsf{out}}(\mathsf{aux})}(x, 1^\lambda)$.*

We also prove that the same lemma holds for the case where the indices of the corrupted parties are $\{2, \mathsf{out}\}$.

Proof. Let us consider the following adversary $\mathcal{A}^{\mathsf{SA}}_{1,\mathsf{out}}$, which is rushing in the first round and having oracle access to two different random functions. $\mathcal{A}^{\mathsf{SA}}_{1,\mathsf{out}}$ queries the functions on the first round message from the honest P_2 obtaining the values x_1, r_1, and it also has the input x_{out} and the randomness r_{out} for corrupted party P_{out}. Then, adversary uses x_1, r_1, x_{out}, r_{out} to compute its first-round messages, in particular, it runs $\{\mathtt{frst\text{-}msg}_i(x_i; r_i)\}_{i\in\{1,\mathsf{out}\}}$ to obtain $\{\mathsf{msg}^1_i\}_{i\in\{1,\mathsf{out}\}}$. For the remaining rounds, it acts as per protocol Π' specification.

We argue that in the eye of honest P_2, interacting with $\mathcal{A}^{\mathsf{r}}_{1,\mathsf{out}}$ is indistinguishable from interacting with $\mathcal{A}^{\mathsf{SA}}_{1,\mathsf{out}}$. The reason behind is that the honest party does not rewind the adversary.

It can not tell whether the adversary use a random input or not. At the same time, when the input is randomly sampled, it is indistinguishable from the output of a random function.

We will prove then that the honest P_2 still completes the protocol when $\mathcal{A}^{\mathsf{SA}}_{1,\mathsf{out}}$ has access to PRFs instead of a random functions. In order to do so, we define a set of 3 hybrids $\{\mathcal{H}_j\}_{j\in[3]}$ where in $\mathcal{H}_1$ the adversary $\mathcal{A}^{\mathsf{SA}}_{1,\mathsf{out}}$ is executed, and in each subsequent hybrid, we substitute and invocation of the random function with a PRF computation. The indistinguishability between the two consecutive hybrids follows from the security of PRF. More in detail, assuming there exists a distinguisher $\mathcal{D}$ that can distinguish between experiments $\mathcal{H}_1$ and $\mathcal{H}_2$, then we can use the following procedure to build a distinguisher for PRF:

- Query the challenger of PRF with first round of the honest P_2 in Π, and receive back a value (either the output of a random function or the output of the PRF), and parse it as the input x_1.
- Compute $\{\texttt{frst-msg}_i(x_i; r_i)\}_{i\in\{1,\mathsf{out}\}}$ to obtain $\{\mathsf{msg}^1_i\}_{i\in\{1,\mathsf{out}\}}$, where x_{out}, r_1, r_{out} are generated as $\mathcal{H}_1$ (respectively as $\mathcal{H}_2$).
- Run the protocol Π', use the message $\{\mathsf{msg}^1_i\}_{i\in\{1,\mathsf{out}\}}$ as the first round messages from $\mathcal{A}^{\mathsf{SA}}_{1,\mathsf{out}}$, and finish the protocol by using $\mathcal{A}^{\mathsf{SA}}_{1,\mathsf{out}}$ (which acts identically in $\mathcal{H}_1$ and $\mathcal{H}_2$).
- Use $\mathcal{D}$ on the transcripts of the protocol, and output what $\mathcal{D}$ outputs.

We now observe that if the challenger provides the output of the random function, we are in $\mathcal{H}_1$, otherwise, we are in $\mathcal{H}_2$. This implies $\mathcal{H}_1 \approx \mathcal{H}_2$. The proofs of the indistinguishability for the other hybrids work similarly.

The proof for the corruption pattern $\{2, \mathsf{out}\}$ is nearly identical to the proof above.

□

We then prove the following lemma, which in summary states that Π remains secure even against such PRF adversaries.

Lemma 15. *Let f be an efficiently computable three-party function. Assume that there exists a three-party protocol Π that securely computes f with selective-abort security when parties P_1 and P_{out} are corrupted, for every PPT real-world adversary $\mathcal{A}^{\mathsf{SA}}_{1,\mathsf{out}}$ with auxiliary input aux. Then for the same corruption pattern $\mathcal{I}$, for the same auxiliary input aux, for all $x \in (\{0,1\}^*)^3$, for all $\lambda \in \mathbb{N}$, it holds that $\{\texttt{REAL}_{\Pi,\mathcal{I},\mathcal{A}^{\mathtt{PRF}}_{1,\mathsf{out}}(\mathsf{aux})}(x, 1^\lambda)\} \stackrel{c}{\equiv} \{\texttt{IDEAL}^{\mathsf{sa\text{-}abort}}_{f,\mathcal{I},\mathcal{S}_{1,\mathsf{out}}(\mathsf{aux})}(x, 1^\lambda)\}$. We also prove that it works for the corruption pattern $\{2, \mathsf{out}\}$*

Proof. Because Π is SA secure against any adversary corrupting parties P_1 and P_{out}, therefore Π' is secure also against $\mathcal{A}^{\mathtt{PRF}}_{1,\mathsf{out}}$.

The proof for the corruption pattern $\{2, \mathsf{out}\}$ similar to the proof above.

□

Step 4: Constructing an adversary that breaks the SA security of Π'. We prove that there exists an adversary $\mathcal{A}'_{\mathsf{SA}}$ that can use the simulator $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ to extract the input from an honest P'_1. This would contradict the SA security of Π'. This adversary $\mathcal{A}'_{\mathsf{SA}}$ (formally described in Figure 5.6) controls the parties P'_2 and P'_{out} and runs internally the simulator $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ [19]. Note that $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ expects to interact with an adversary which corrupts P'_1 and P'_{out} in an execution of Π', hence $\mathcal{A}'_{\mathsf{SA}}$ needs to make sure that $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ can be executed correctly, despite the party P'_1

[19] Note that the simulator is expected polynomial time, hence we need to cut its running time to make sure that $\mathcal{A}'_{\mathsf{SA}}$ remains PPT.

being honest. In particular, we need to argue that the simulator can work properly (i.e., the simulator extracts the input P_1') while P_1' is not rewound.

Figure 5.6: The adversary $\mathcal{A}'_{\mathsf{SA}}$

- Define and initialize $j \leftarrow -1$ and sample $k_1, k_2 \leftarrow \{0,1\}^\lambda$. Run $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$, which denotes the simulator of Π' for the case where P_1' and P'_{out} are corrupted. $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ is run until it performs up to κ steps (recall that $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ needs $\kappa = \mathsf{poly}(\lambda)$ expected number steps).
- Sample $i \leftarrow [\kappa]$. Any time that $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ opens a new session by sending a new first-round m (m here denotes all the messages received over broadcast in the first round) to the honest P_1' then set $j \leftarrow j+1$ and do the following.
  - If $j = i$ compute $x_1^i \leftarrow \mathtt{F}_{k_1}(m)$, where x_1^i will denote the input of the honest party P_1' used in the MPC indistinguishability game[a].
  - If $j \neq i$ then reply to all the queries of $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$, acting as the honest P_1' would act using the input x_1^j and the randomness r_1^j, where $x_1^j \leftarrow \mathtt{F}_{k_1}(m)$ and $r_1^j \leftarrow \mathtt{F}_{k_2}(m)$.
- In the i-th session act as a proxy between the MPC challenger and $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ with respect to all the messages related to P_1'. Note that the messages from the challenger will either be simulated or generated by running the honest party P_1' with the input x_1^i.
- For every message that $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ sends to P_1' in the session $j \neq i$, reply as the honest party P_1' would using the input x_1^j and the randomness r_1^j.
- Act as an honest P'_{out} would act with the only difference that P'_{out} sends the third round to P_1' in the session i only after that $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ has stopped and it has returned a transcript consistent with the i-th session.
- Whenever $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ tries to send a second round in the i-th session, forward this message only to P'_{out}. When $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ stops and returns its transcript, forward to P_1' the second round message that appears in the transcript.
- When $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ attempts to query the ideal functionality[b] with a value $\tilde{x}_1 = (\tilde{x}^0, \tilde{x}^1)$, $\mathcal{A}'_{\mathsf{SA}}$ records this value, and sends back to the simulator $\tilde{x}^{x_2}$ (here the adversary acts as the ideal functionality would for the simulator $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$). When $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ stops, and returns its output, check if the output transcript is consistent with the messages generated in the i-th session. If this is the case then do the following
  - If $\tilde{x}_1 = x_1^i$ then return 1 (this is to denote that the challenger generated a transcript using the honest procedure for P_1').
  - If $\tilde{x}_1 \neq x_1^i$ then return 0 (this is to denote that the messages computed by the challenger on behalf of P_1' were simulated)

  If instead the output transcript of $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ is not consistent with the i-th session, then return a random bit.

[a] Note that in the security experiment of MPC protocol must hold for any x_1. In particular, this means that the security must hold for a value x_1 chosen by the adversary prior to the beginning of the experiment.

[b] We consider the oblivious transfer functionality, where P_1''s input is $x_1 = (x^0 \in \{0,1\}^\lambda, x^1 \in \{0,1\}^\lambda)$ and P_2''s input is $x_2 \in \{0,1\}$, while P'_{out} does not have any input. We are going to explain why we choose this functionality later.

In other words, we need to prove that any rewind made by the simulator $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ can be emulated by $\mathcal{A}'_{\mathsf{SA}}$ without rewinding the honest P_1'. Finally note that P'_{out} is corrupted, hence $\mathcal{A}'_{\mathsf{SA}}$ can emulate any interaction between $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ and P'_{out}. This means that we can rewind P'_{out}, but at the same time need to make sure that any rewind performed on P'_{out}, does not implicitly rewind also P_1'. To make sure this does not happen, we instruct $\mathcal{A}'_{\mathsf{SA}}$ to be rushing in the first round and non-rushing in other rounds. We examine now all the possible rewinding patterns, to show that our adversary does not need to rewind P_1'. We do so using different figures (for each pattern), where we use a straight line to denote messages on the $P2P$ channel, a dashed line to denote messages on the broadcast channel and a dotted line to denote rewind messages and corresponding new messages. We also use the number to indicate which round message it is.

We start by considering the scenario where $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ is attempting to rewind (what he sees as a malicious) P_1' in the first BC round. Because we cut the running time of $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$, we can assume that $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ will rewind the first BC round for at most κ times, for some polynomial κ. To deal with this situation, our adversary samples a random index $i \leftarrow [\kappa]$, and forwards to the external challenger only the message generated by $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ related to the i-th session. To define the input x_1 that will be used in the indistinguishability experiment, we evaluate a PRF on input the first round messages received from $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$. Note that the security of Π' must hold for any choice of x_1, even for an adversarially chosen one.

For all the other sessions, our adversary will answer the messages generated by $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ acting as the honest P_1' would, using as input and randomness the output of a PRF evaluated on the messages received from $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$. By applying Lemma 15, we can argue that $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ is secure against $\mathcal{A}^{\mathtt{PRF}}$, which means even when the input and randomness are computed by using the PRF, the simulator is still able to extract the input from the corrupted party, with non-negligible probability.

We mentioned that in the i-th session, $\mathcal{A}'_{\mathsf{SA}}$ acts as a proxy between the external challenger and $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ for all the messages related to P_1'. Note that $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ can also rewind the second, third, and fourth rounds in the i-th session. Consider the case where $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ rewinds the second round that goes from P_2' to P'_{out}. This action, in turn, causes P'_{out} to send multiple third rounds to P_1'. We observe that there is no need to forward all these multiple third rounds to P_1', as we can just block all of them except the message that will appear in the final simulated session. Note that the simulator must work well even with this modification since the simulator does not see the effect of the rewinds implicitly performed to P_1' due to the fact that the simulator has no access to the fourth $P2P$ round messages that P_1' may compute as a consequence of these rewinds (recall that P'_{out} is non-rushing and that it does not send any message to P_2'). We refer to Figure 5.7 for a pictorial description.

Figure 5.7: Pattern 2: $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ rewinds the second BC round

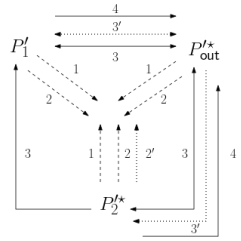


Figure 5.8: Pattern 3: $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ rewinds the third $P2P$ round for P'_1

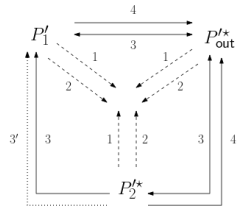
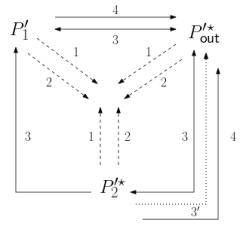


Figure 5.8 reflects the scenario where $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ is attempting to rewind (what he sees as a malicious) P'_1 in the third $P2P$ round. By the definition of Π', P'_1 does not react on the third round that comes from P'_1, hence, we can just forward to P'_1 only one message, which corresponds to the message the simulator returns in its final simulated transcript.

Figure 5.9: Pattern 4: $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ rewinds the third $P2P$ round for P'_{out}



reflects the scenario where $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ is attempting a rewind to P'_{out} in the third $P2P$ round. By construction of Π', P'_{out} does not send any fourth message. Hence, we can simply allow this rewind as the adversary $\mathcal{A}'_{\mathsf{SA}}$ controls P'_{out}.

Figure 5.10: Pattern 5: $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ rewinds the fourth $P2P$ round for P'_{out}

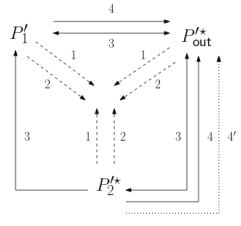


Figure 5.10 reflects the scenario where $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ is attempting a rewind P'_{out} in the fourth $P2P$ round. Also in this case, it is easy to see that this action does not implicitly rewind P'_1.

We have argued our adversary can run $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$, without perturbing its behavior, while at the same time making sure that P'_1 is not rewound. This means that the $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ will, with non-

negligible probability, return some value that corresponds to the input of P_1' (when the messages of P_1' are computed accordingly to Π', and are not simulated by the external challenger). Note that $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$ may complete a session, where P_1' is fully under the control of the adversary. This happens when the simulated transcript corresponds to some session $i' \neq i$. However, we can argue that with non-negligible probability, the simulated transcript returned by $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$ does correspond to the i-th session. Once we have argued that, we can claim that $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$ will return the input of P_1', when the MPC challenger computes the messages on behalf of P_1' using the input x_1^i. To reach a contradiction, we need to consider a function that does not implicitly leak the input of P_1' to the adversary. For this, we consider the oblivious transfer functionality, where P_1''s input is $x_1 = (x^0 \in \{0,1\}^\lambda, x^1 \in \{0,1\}^\lambda)$ and P_2''s input is $x_2 \in \{0,1\}$, while P_{out}' does not have any input.:

$$f(x_1, x_2, \perp) = x^{x_2}$$

Finally, given that the probability that $\mathcal{A}_{\mathsf{SA}}'$ guess the session i correctly is non-negligible, and given that the simulator run internally by $\mathcal{A}_{\mathsf{SA}}'$ will succeed with non-negligible probability, we can claim that our adversary $\mathcal{A}_{\mathsf{SA}}'$ breaks the SA security Π' with non-negligible advantage.

Our theorem can be extended to n-party cases. Assuming there exists a n-party four-round protocol that can compute f with selective identifiable-abort security, against $t < n$ corruptions. We denote the n parties running the protocol with $(P_1, \ldots, P_n)$, then we let P_1 take the input x_1, P_2 take the input x_2, and other parties take no input. If such a protocol would be secure, then we can easily construct a 3-party protocol (where all the parties that have no input are emulated by a single entity) to compute f with selective identifiable-abort security, which would contradict our claim.

□

Theorem 16 (*BC-BC-P2P-BC*, SIA, Plain Model, $n > t$)**.** *Assume the existence of pseudo-random functions. There exists function f such that no n-party four-round protocol (in the plain model) can compute f with selective identifiable-abort security, against $t < n$ corruptions, while in the protocol, the first two rounds use broadcast channels and the third one uses peer-to-peer channels and the last round uses broadcast channels.*

Proof. The proof is similar to the proof of Theorem 12 with some differences in Step 1 and Step 2; detail follows.

By contradiction, assume that there exists a three-party protocol Π that can compute any function f with selective identifiable-abort security. Let us denote the three parties running the protocol Π with $(P_1, P_2, P_{\mathsf{out}})$. We proceed now in a series of steps in order to reach a contradiction.

Step 1: $\mathbf{P_{out}}$ can not abort and identify the corrupted party. This step follows from Lemma 13 which implies that P_{out} must obtain the output.

Step 2: Constructing a SA secure protocol (only for some corruption patterns). We prove that if Π is SIA secure, then we can construct a new protocol Π' that is SA secure, for some particular corruption patterns, namely when parties P_1' and P_{out}' are corrupted or when parties P_2' and P_{out}' are corrupted. We denote three parties running the protocol Π' with $(P_1', P_2', P_{\mathsf{out}}')$.

The protocol Π' that we construct in this proof is the same as the one of Figure 5.3. Note that since only P_{out}' obtains output, the last round can be replaced by using $P2P$ channels. More specifically, in Π' of Figure 5.3, in the last round, P_1' and P_2' only send messages to P_{out}',

and P'_{out} does not send any message. Therefore, these messages can be sent also privately to P'_{out} since we are restricted to prove that Π' satisfies SA security.

The proofs of the remaining steps are exactly identical to Theorem 12.

□

Theorem 17 (*P2P-BC-BC-BC*, SIA, Plain Model, $n > t$)**.** *Assume the existence of pseudo-random functions. There exists function f such that no n-party four-round protocol (in the plain model) can compute f with selective identifiable-abort security, against $t < n$ corruptions, while in the protocol, the first round uses the peer-to-peer channels and the remaining three rounds use broadcast channels.*

Proof. The proof of this theorem is similar to the proof of Theorem 12. Here, we follow the same proof approach, and we only formally describe the part that has major differences.

By contradiction, assume that there exists a three-party protocol Π that can compute any function f with selective identifiable-abort security. Let denote the three parties running the protocol Π with $(P_1, P_2, P_{\mathsf{out}})$. We proceed now in a series of steps in order to reach a contradiction.

Step 1: $\mathbf{P_{out}}$ can not abort and identify the corrupted party. This step follows from Lemma 13 which implies that P_{out} must obtain the output.

Step 2: Constructing a SA secure protocol (only for some corruption patterns). We prove that if Π is SIA secure, then we can construct a new protocol Π' (in Figure 5.11) which is SA secure for two corruption patterns. Applying the same reasoning of Theorem 16, we focus on the case where the last round of Π' is over $P2P$ channel.

This protocol Π' (executed between parties P'_1, P'_2 and P'_{out}) works exactly like Π, with the following differences:

- The honest P'_1 does not send the third message to P'_2.
- No fourth messages between P'_1 and P'_2.
- P'_{out} does not send any fourth round to P'_1 and P'_2.

We prove that this protocol is secure with selective abort. Informally, this is possible because the honest parties send fewer messages compared to Π, and the party P'_{out} will still be able to compute the output due to the argument given above. We proceed more formally below.

Figure 5.11: The new protocol Π'

Primitives: A three-party four-round protocol $\Pi = \{(\texttt{frst-msg}_i, \{\texttt{nxt-msg}_i^k\}_{k\in\{1,2,3\}}, \texttt{output}_i)\}_{i\in\{1,2,\mathsf{out}\}}$ that securely computes any f with selective identifiable-abort security against $t < n$ corruptions, where the first round uses the $P2P$ channels to exchange messages, and last three rounds use the broadcast channels.
Private input: Every party P'_i has a private input $x_i \in \{0,1\}^*$.

First round (P2P):

1. Every party P'_i samples the randomness r_i from uniform distribution $\mathcal{D}$, computes $(\{\mathsf{msg}^1_{i\to j}\}_{j\in\{1,2,\mathsf{out}\}}) \leftarrow \texttt{frst-msg}_i(x_i; r_i)$

2. P'_1 sends $\mathsf{msg}^1_{1\to\mathsf{out}}$ to P'_{out}, P'_{out} sends $\mathsf{msg}^1_{\mathsf{out}\to 1}$ to P'_1, and sends $\mathsf{msg}^1_{\mathsf{out}\to 2}$ to P'_2. P'_2 sends $\mathsf{msg}^1_{2\to 1}$ to P'_1, and sends $\mathsf{msg}^1_{2\to\mathsf{out}}$ to P'_{out}.

Second round (BC):

1. P'_1 sets $\mathsf{msg}^1_{2\to 1} = \bot$, and computes $\mathsf{msg}^2_1 \leftarrow \mathtt{nxt\text{-}msg}^1_1(x_1, \{\mathsf{msg}^1_{j\to 1}\}_{j\in\{1,2,\mathsf{out}\}}; r_1)$, P'_{out} computes $\mathsf{msg}^2_{\mathsf{out}} \leftarrow \mathtt{nxt\text{-}msg}^1_{\mathsf{out}}(x_{\mathsf{out}}, \{\mathsf{msg}^1_{j\to\mathsf{out}}\}_{j\in\{1,2,\mathsf{out}\}}; r_{\mathsf{out}})$, P'_2 computes $\mathsf{msg}^2_2 \leftarrow \mathtt{nxt\text{-}msg}^1_2(x_2, \{\mathsf{msg}^1_{j\to 2}\}_{j\in\{2,\mathsf{out}\}}; r_2)$.

2. Every party P'_i sends the message over the broadcast channel.

Third round (BC):

1. P'_1 computes $\mathsf{msg}^3_1 \leftarrow \mathtt{nxt\text{-}msg}^2_1(x_1, \{\mathsf{msg}^1_{j\to 1}\}_{j\in\{1,2,\mathsf{out}\}}, \{\mathsf{msg}^2_j\}_{j\in\{1,2,\mathsf{out}\}}; r_1)$, P'_{out} computes $\mathsf{msg}^3_{\mathsf{out}} \leftarrow \mathtt{nxt\text{-}msg}^2_{\mathsf{out}}(x_{\mathsf{out}}, \{\mathsf{msg}^1_{j\to\mathsf{out}}\}_{j\in\{1,2,\mathsf{out}\}}, \{\mathsf{msg}^2_j\}_{j\in\{1,2,\mathsf{out}\}}; r_{\mathsf{out}})$, P'_2 computes $\mathsf{msg}^3_2 \leftarrow \mathtt{nxt\text{-}msg}^2_2(x_2, \{\mathsf{msg}^1_{j\to 2}\}_{j\in\{2,\mathsf{out}\}}, \{\mathsf{msg}^2_j\}_{j\in\{1,2,\mathsf{out}\}}; r_2)$.

2. Every party P'_i sends the message over the broadcast channel.

Fourth round (P2P):

1. P'_1 computes $\mathsf{msg}^4_1 \leftarrow \mathtt{nxt\text{-}msg}^3_1(x_1, \{\mathsf{msg}^1_{j\to 1}\}_{j\in\{1,2,\mathsf{out}\}}, \{\mathsf{msg}^k_j\}_{j\in\{1,2,\mathsf{out}\},k\in\{2,3\}}; r_1)$, P'_{out} computes $\mathsf{msg}^4_{\mathsf{out}} \leftarrow \mathtt{nxt\text{-}msg}^3_{\mathsf{out}}(x_{\mathsf{out}}, \{\mathsf{msg}^1_{j\to\mathsf{out}}\}_{j\in\{1,2,\mathsf{out}\}}, \{\mathsf{msg}^k_j\}_{j\in\{1,2,\mathsf{out}\},k\in\{2,3\}}; r_{\mathsf{out}})$, P'_2 computes $\mathsf{msg}^4_2 \leftarrow \mathtt{nxt\text{-}msg}^3_2(x_2, \{\mathsf{msg}^1_{j\to 2}\}_{j\in\{2,\mathsf{out}\}}, \{\mathsf{msg}^k_j\}_{j\in\{1,2,\mathsf{out}\},k\in\{2,3\}}; r_2)$.

2. P'_1 sends msg^4_1 to P'_{out}, P'_2 sends msg^4_2 to P'_{out}.

Output Computation:

1. P'_{out} compute and output $y \leftarrow \mathtt{output}_{\mathsf{out}}(x_{\mathsf{out}}, \{\mathsf{msg}^1_{j\to\mathsf{out}}\}_{j\in\{1,2,\mathsf{out}\}}, \{\mathsf{msg}^k_j\}_{j\in\{1,2,\mathsf{out}\},k\in\{2,3,4\}}; r_{\mathsf{out}})$

We now prove that Π' is secure when 1) P'_1 and P'_{out} are corrupted or when 2) P'_2 and P'_{out} are corrupted. In order to do so we consider the simulators of Π $\mathcal{S}^{\mathsf{SIA}}_{1,\mathsf{out}}$ (for corrupted P_1 and P_{out}), and $\mathcal{S}^{\mathsf{SIA}}_{2,\mathsf{out}}$ (for corrupted P_2 and P_{out}) and we construct to new simulators $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ and $\mathcal{S}^{\mathsf{SA}}_{2,\mathsf{out}}$ which will be used to prove the security of Π' in the above mentioned corruption patterns. In order to do so we build now an adversary $\mathcal{M}_{\mathsf{intf}}$ (Figure 5.12) which acts as an adversary of Π (and does so internally running $\mathcal{A}^{\mathsf{SA}}$) for $\mathcal{S}^{\mathsf{SIA}}_{1,\mathsf{out}}$ (res. $\mathcal{S}^{\mathsf{SIA}}_{2,\mathsf{out}}$). In this, we denote as *left interface* the interface where the adversary sends and receives the protocol messages.

Figure 5.12: The adversary $\mathcal{M}_{\mathsf{intf}}$

Notation: Let $\mathbb{H}$ be the set of indices of the honest parties and $\mathcal{I}$ be the indices of the corrupted parties. $\mathcal{M}_{\mathsf{intf}}$ internally runs the adversary $\mathcal{A}^{\mathsf{SA}}$, and is equipped with a left interface, where it receives the messages computed on behalf of the honest parties and sends the messages computed on the behalf of the corrupted parties.

First round (P2P):

1. Upon receiving $\mathsf{msg}^1_{h\to j}$ where $h \in \mathbb{H}$ and $j \in \mathcal{I}$ from the left interface, $\mathcal{M}_{\mathsf{intf}}$ forwards the message $\mathsf{msg}^1_{h\to\mathsf{out}}$ (and the message $\mathsf{msg}^1_{2\to 1}$ in the case where $2 \in \mathbb{H}$)

to $\mathcal{A}^{\mathsf{SA}}$ in the execution of Π'.

2. Upon receiving the messages sent by $\mathcal{A}^{\mathsf{SA}}$, $\mathcal{M}_{\mathsf{intf}}$ forwards them to the left interface, where it is acting as a corrupted party for Π.

Second round (BC):

1. Upon receiving msg_h^2 from the left interface where $h \in \mathbb{H}$, $\mathcal{M}_{\mathsf{intf}}$ forwards the message to $\mathcal{A}^{\mathsf{SA}}$ in the execution of Π'.

2. Upon receiving the messages sent by $\mathcal{A}^{\mathsf{SA}}$, $\mathcal{M}_{\mathsf{intf}}$ forwards them to the left interface, where it is acting as a corrupted party for Π.

Third round (BC):

1. Upon receiving msg_h^3 from the left interface where $h \in \mathbb{H}$, $\mathcal{M}_{\mathsf{intf}}$ forwards the message to $\mathcal{A}^{\mathsf{SA}}$ in the execution of Π'.

2. Upon receiving the messages sent by $\mathcal{A}^{\mathsf{SA}}$, $\mathcal{M}_{\mathsf{intf}}$ forwards them to the left interface, where it is acting as a corrupted party for Π.

Fourth round (P2P):

1. Upon receiving msg_h^4 where $h \in \mathbb{H}$ from the left interface, $\mathcal{M}_{\mathsf{intf}}$ forwards the message msg_h^4 to $\mathcal{A}^{\mathsf{SA}}$ in the execution of Π'.

2. Upon receiving the messages sent by $\mathcal{A}^{\mathsf{SA}}$, $\mathcal{M}_{\mathsf{intf}}$ forwards them to the left interface, where it is acting as a corrupted party for Π.

Then we formally describe $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$ in Figure 5.13.

Figure 5.13: $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$

$\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$ performs the following steps:

- Invoke $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SIA}}$ for the adversary $\mathcal{M}_{\mathsf{intf}}$.
- Work as a proxy between the trusted party and $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SIA}}$.

In the end, $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SA}}$ output whatever $\mathcal{S}_{1,\mathsf{out}}^{\mathsf{SIA}}$ outputs, and halt.

We can construct the simulator $\mathcal{S}_{2,\mathsf{out}}^{\mathsf{SA}}$ by using $\mathcal{S}_{2,\mathsf{out}}^{\mathsf{SIA}}$ similarly. The remaining proof is identical to this part of the proof in Theorem 12.

After we prove the SA security of Π', we are going to show how to construct a SA adversary $\mathcal{A}'_{\mathsf{SA}}$ to break the SA security of Π' to reach a contradiction. Note that Step 3 works similarly to the proof of Theorem 12, with the only difference that the adversarial P'_1 is also rushing in the second round (while is non-rushing in the remaining 3rd and 4th round) and applies the PRF to compute the second round to P'_2 of the protocol Π' (recall P'_1 does not send a first-round message to P'_2).

Step 4: Constructing a specific adversary that breaks the SA security of Π'. $\mathcal{A}'_{\mathsf{SA}}$ uses nearly the same procedure of Figure 5.6, and the invoked simulator is the one of Figure 5.13.

The difference is that (in the internal emulated session) $\mathcal{A}'_{\mathsf{SA}}$ computes the second round between P'_1 and P'_2 by applying the PRF on the new first and second round sent by $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ (on behalf of P'_2).

Crucially, we need to argue that when the adversary $\mathcal{A}'_{\mathsf{SA}}$ corrupts parties P'_2 and P'_{out}, then it can run $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ emulating for her an execution of Π' were parties P'_1 and P'_{out} are the one corrupted. It is very important to argue that in this execution the rewinds made by $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ (acting as P'_2) are not useful for extracting the input of P'_1, since $\mathcal{A}'_{\mathsf{SA}}$ is interacting with an honest P'_1 and can not perform rewind on an honest player. In other words, any rewind made by the simulator $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ can be emulated by $\mathcal{A}'_{\mathsf{SA}}$ without rewinding the honest P'_1. Finally note that P'_{out} is corrupted so $\mathcal{A}'_{\mathsf{SA}}$ can emulate any interaction between $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ and P'_{out}, so we only analyze when this interaction can indirectly rewind P'_1.

We examine now all the possible rewinding patterns using the same notation as in Theorem 12.

- **Pattern 1:** $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ rewinds the first $P2P$ round for P'_1. In Pattern 1, there is no new message generated by P'_1, since based on Figure 5.11, P'_1 ignores the first round message from P'_2 (and therefore from the simulator acting as P'_{out}). We can conclude that this pattern can be emulated by $\mathcal{A}'_{\mathsf{SA}}$ while interacting with $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$.

- **Pattern 2:** $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ rewinds the first $P2P$ round for P'_{out}, this follows with a similar argument to the one explained for Pattern 1 in Theorem 12.

- **Pattern 3:** $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ rewinds the second BC round. First observe that Lemma 15 (with minor modification) still holds, and the proof follows similar arguments. Therefore this pattern follows with a similar argument to the one explained for Pattern 1 in Theorem 12. Roughly speaking $\mathcal{A}'_{\mathsf{SA}}$ can prepare required messages for the simulator emulating all the sessions internally except for the i-th in which is interacting with the honest P'_1 without rewinding it.

- **Pattern 4:** $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ rewinds the third BC round. Since, by description of Figure 5.11, there is no incoming 4th round message for P'_2 (and therefore from the simulator acting as P'_2), we can conclude that this pattern can be emulated by $\mathcal{A}'_{\mathsf{SA}}$ while interacting with $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$.

- **Pattern 5:** $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ rewinds the fourth $P2P$ round (which is sent only to P'_{out}), this follows from the same arguments explained for Pattern 5 in Theorem 12.

Therefore, we can conclude that all the patterns can be emulated by $\mathcal{A}'_{\mathsf{SA}}$ while interacting with $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$.

By using the same ideal functionality and following the same procedures in Theorem 12, we reach a contradiction, and this result can be extended to n-party cases similar to Theorem 12. □

Theorem 18 (*BC-P2P-BC-BC*, SIA, Plain Model, $n > t$)**.** *Assume the existence of pseudo-random functions. There exists function f such that no n-party four-round protocol (in the plain model) can compute f with selective identifiable-abort security, against $t < n$ corruptions, while in the protocol, the first round uses broadcast channels and the second one uses peer-to-peer channels and the last two rounds use broadcast channels.*

Proof. For this theorem, we also follow the same proof strategy of Theorem 12.

By contradiction, assume that there exists a three-party protocol Π that can compute any function f with selective identifiable-abort security. Let denote the three parties running the

protocol Π with $(P_1, P_2, P_{\mathsf{out}})$. We proceed now in a series of steps in order to reach a contradiction.

Step 1: $\mathbf{P_{out}}$ can not abort and identify the corrupted party. This step follows from Lemma 13 which implies that P_{out} must obtain the output.

Step 2: Constructing a SA secure protocol (only for some corruption patterns). We prove that if Π is SIA secure, then we can construct a new protocol Π' (in Figure 5.14) which is SA secure for two corruption patterns. Applying the same reasoning of Theorem 16, we focus on the case where the last round of Π' is over $P2P$ channel.

This protocol Π' (executed between parties P'_1, P'_2 and P'_{out}) works exactly like Π, with the following differences:

- The honest P'_1 does not send the third message to P'_2.
- No fourth messages between P'_1 and P'_2.
- P'_{out} does not send any fourth round to P'_1 and P'_2.

We prove that this protocol is secure with selective aborts. Informally, this is possible because the honest parties send fewer messages compared to Π, and the party P'_{out} will still be able to compute the output due to the argument given above. We proceed more formally below.

Figure 5.14: The new protocol Π'

Primitives: A three-party four-round protocol $\Pi = \{(\texttt{frst-msg}_i, \{\texttt{nxt-msg}_i^k\}_{k\in\{1,2,3\}}, \texttt{output}_i)\}_{i\in\{1,2,3\}}$ that securely computes any f with selective identifiable-abort security against $t < n$ corruptions, where the first round uses the broadcast channels to exchange messages, the second round use the $P2P$ channels, and last two rounds use the broadcast channels.
Private input: Every party P'_i has a private input $x_i \in \{0,1\}^*$.

First round (BC):
Every party P'_i samples the randomness r_i from uniform distribution $\mathcal{D}$, computes $\mathsf{msg}_i^1 \leftarrow \texttt{frst-msg}_i(x_i; r_i)$, and sends the message over the broadcast channel.
Second round (P2P):

1. Every party P'_i computes $\{\mathsf{msg}^2_{i\to j}\}_{j\in\{1,2,\mathsf{out}\}} \leftarrow \texttt{nxt-msg}_i^1(x_i, \{\mathsf{msg}_j^1\}_{j\in\{1,2,\mathsf{out}\}}; r_i)$.
2. P'_1 sends $\mathsf{msg}^2_{1\to\mathsf{out}}$ to P'_{out}, P'_{out} sends $\mathsf{msg}^2_{\mathsf{out}\to 1}$ to P'_1, and sends $\mathsf{msg}^2_{\mathsf{out}\to 2}$ to P'_2. P'_2 sends $\mathsf{msg}^2_{2\to 1}$ to P'_1, and sends $\mathsf{msg}^2_{2\to\mathsf{out}}$ to P'_{out}.

Third round (BC):

1. P'_1 sets $\mathsf{msg}^2_{2\to 1} = \bot$, and computes $\mathsf{msg}_1^3 \leftarrow \texttt{nxt-msg}_1^2(x_1, \{\mathsf{msg}_j^1\}_{j\in\{1,2,\mathsf{out}\}}, \{\mathsf{msg}^2_{j\to 1}\}_{j\in\{1,2,\mathsf{out}\}}; r_1)$, P'_{out} computes $\mathsf{msg}^3_{\mathsf{out}} \leftarrow \texttt{nxt-msg}^2_{\mathsf{out}}(x_{\mathsf{out}}, \{\mathsf{msg}_j^1\}_{j\in\{1,2,\mathsf{out}\}}, \{\mathsf{msg}^2_{j\to 2}\}_{j\in\{1,2,\mathsf{out}\}}; r_2)$, P'_2 computes $\mathsf{msg}_2^3 \leftarrow \texttt{nxt-msg}_2^2(x_2, \{\mathsf{msg}_j^1\}_{j\in\{1,2,\mathsf{out}\}}, \{\mathsf{msg}^2_{j\to 2}\}_{j\in\{2,\mathsf{out}\}}; r_2)$.
2. Every party P'_i sends the computed message over the broadcast channel.

Fourth round (P2P):

1. P'_1 computes $\mathsf{msg}^4_1 \leftarrow \mathtt{nxt\text{-}msg}^3_1(x_1, \{\mathsf{msg}^k_j\}_{j\in\{1,2,\mathsf{out}\},k\in\{1,3\}}, \{\mathsf{msg}^2_{j\to 1}\}_{j\in\{1,2,\mathsf{out}\}}; r_1)$, P'_{out} computes $\mathsf{msg}^4_{\mathsf{out}} \leftarrow \mathtt{nxt\text{-}msg}^3_{\mathsf{out}}(x_{\mathsf{out}}, \{\mathsf{msg}^k_j\}_{j\in\{1,2,\mathsf{out}\},k\in\{1,3\}}, \{\mathsf{msg}^2_{j\to\mathsf{out}}\}_{j\in\{1,2,\mathsf{out}\}}; r_{\mathsf{out}})$, P'_2 computes $\mathsf{msg}^4_2 \leftarrow \mathtt{nxt\text{-}msg}^3_2(x_2, \{\mathsf{msg}^k_j\}_{j\in\{1,2,\mathsf{out}\},k\in\{1,3\}}, \{\mathsf{msg}^2_{j\to 2}\}_{j\in\{2,\mathsf{out}\}}; r_2)$.

2. P'_1 sends msg^4_1 to P'_2, P'_3 sends msg^4_3 to P'_2.

Output Computation:

1. P'_{out} compute and output $y \leftarrow \mathtt{output}_{\mathsf{out}}(x_{\mathsf{out}}, \{\mathsf{msg}^k_j\}_{j\in\{1,2,\mathsf{out}\},k\in\{1,3,4\}}, \{\mathsf{msg}^2_{j\to\mathsf{out}}\}_{j\in\{1,2,\mathsf{out}\}}; r_{\mathsf{out}})$

We now prove that Π' is secure when 1) P'_1 and P'_{out} are corrupted or when 2) P'_2 and P'_{out} are corrupted. In order to do so we consider the simulators of Π $\mathcal{S}^{\mathsf{SIA}}_{1,\mathsf{out}}$ (for corrupted P_1 and P_{out}), and $\mathcal{S}^{\mathsf{SIA}}_{2,\mathsf{out}}$ (for corrupted P_2 and P_{out}) and we construct to new simulators $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ and $\mathcal{S}^{\mathsf{SA}}_{2,\mathsf{out}}$ which will be used to prove the security of Π' in the above mentioned corruption patterns. In order to do so we build now an adversary $\mathcal{M}_{\mathsf{intf}}$ (Figure 5.15) which acts as an adversary of Π (and does so internally running $\mathcal{A}^{\mathsf{SA}}$) for $\mathcal{S}^{\mathsf{SIA}}_{1,\mathsf{out}}$ (res. $\mathcal{S}^{\mathsf{SIA}}_{2,\mathsf{out}}$). Let us denote with *left interface*, the interface where the adversary sends and receives the protocol messages.

Figure 5.15: The adversary $\mathcal{M}_{\mathsf{intf}}$

Notation: Let $\mathbb{H}$ be the set of indices of the honest parties and $\mathcal{I}$ be the indices of the corrupted parties. $\mathcal{M}_{\mathsf{intf}}$ internally runs the adversary $\mathcal{A}^{\mathsf{SA}}$, and is equipped with a left interface, where it receives the messages computed on behalf of the honest parties and sends the messages computed on the behalf of the corrupted parties.

First round (BC):

1. Upon receiving msg^1_h from the left interface where $h \in \mathbb{H}$, $\mathcal{M}_{\mathsf{intf}}$ forwards the message to $\mathcal{A}^{\mathsf{SA}}$ in the execution of Π'.

2. Upon receiving the messages sent by $\mathcal{A}^{\mathsf{SA}}$ in Π', $\mathcal{M}_{\mathsf{intf}}$ forwards them to the left interface, where it is acting as a corrupted party for Π.

Second round (P2P):

1. Upon receiving $\mathsf{msg}^2_{h\to j}$ where $h \in \mathbb{H}$ and $j \in \mathcal{I}$ from the left interface, $\mathcal{M}_{\mathsf{intf}}$ forwards the message $\mathsf{msg}^2_{h\to\mathsf{out}}$ (and the message $\mathsf{msg}^2_{2\to 1}$ in the case where $2 \in \mathbb{H}$) to $\mathcal{A}^{\mathsf{SA}}$ in the execution of Π'.

2. Upon receiving the messages sent by $\mathcal{A}^{\mathsf{SA}}$ in Π', $\mathcal{M}_{\mathsf{intf}}$ forwards them to the left interface, where it is acting as a corrupted party for Π.

Third round (BC):

1. Upon receiving msg^3_h from the left interface where $h \in \mathbb{H}$, $\mathcal{M}_{\mathsf{intf}}$ forwards the message to $\mathcal{A}^{\mathsf{SA}}$ in the execution of Π'.

2. Upon receiving the messages sent by $\mathcal{A}^{\mathsf{SA}}$ in Π', $\mathcal{M}_{\mathsf{intf}}$ forwards them to the left interface, where it is acting as a corrupted party for Π.

Fourth round (P2P):

1. Upon receiving $\mathsf{msg}^4_{h\to\mathsf{out}}$ where $h \in \mathbb{H}$ in the execution of Π, $\mathcal{M}_{\mathsf{intf}}$ forwards the message $\mathsf{msg}^4_{h\to\mathsf{out}}$ to $\mathcal{A}^{\mathsf{SA}}$ in the execution of Π'.

2. Upon receiving the messages sent by $\mathcal{A}^{\mathsf{SA}}$ in Π', $\mathcal{M}_{\mathsf{intf}}$ forwards them to the left interface, where it is acting as a corrupted party for Π.

Then we formally describe $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ in Figure 5.16.

Figure 5.16: $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$

$\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ performs the following steps:

- Invoke $\mathcal{S}^{\mathsf{SIA}}_{1,\mathsf{out}}$ for the adversary $\mathcal{M}_{\mathsf{intf}}$.
- Work as a proxy between the trusted party and $\mathcal{S}^{\mathsf{SIA}}_{1,\mathsf{out}}$.

In a similar way we can construct the simulator $\mathcal{S}^{\mathsf{SA}}_{2,\mathsf{out}}$, but making use of $\mathcal{S}^{\mathsf{SIA}}_{2,\mathsf{out}}$. Proving that Π' enjoys SA security (for the considered corruption patterns) works similar to the proof in Theorem 12.

After we prove the SA security of Π', we are going to show how to construct a SA adversary $\mathcal{A}'_{\mathsf{SA}}$ to break the SA security of Π' to reach a contradiction (Step 3 works similar to the proof of Theorem 12).

Step 4: Constructing a specific adversary that breaks the SA security of Π'. $\mathcal{A}'_{\mathsf{SA}}$ uses the same procedure of Figure 5.6, and the invoked simulator is the one of Figure 5.16. Crucially, we need to argue that when the adversary $\mathcal{A}'_{\mathsf{SA}}$ corrupts parties P'_2 and P'_{out}, then he can run $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ emulating for him an execution of Π' were parties P'_1 and P'_{out} are the one corrupted. It is very important to argue that in this execution the rewinds made by $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ (acting as P'_2) are not useful for extracting the input of P'_1, since $\mathcal{A}'_{\mathsf{SA}}$ is interacting with an honest P'_1 and can not perform rewind on an honest player. In other words, any rewind made by the simulator $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ can be emulated by $\mathcal{A}'_{\mathsf{SA}}$ without rewinding the honest P'_1. Finally note that $P'^{,\star}_{\mathsf{out}}$ is corrupted so $\mathcal{A}'_{\mathsf{SA}}$ can emulate any interaction between $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ and $P'^{,\star}_2$, so we only analyze when this interaction can indirectly rewinding P'_1.

We examine now all the possible rewinding patterns using the same notation as in Theorem 12.

- **Pattern 1:** $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ rewinds the first round. This pattern is equivalent to the Pattern 1 of Theorem 12, and we follow exactly the same proof.
- **Pattern 2:** $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ rewinds the second $P2P$ round for P'_1. In Pattern 2, there is no new message generated by P'_1, since based on Figure 5.14, P'_1 ignores the 2nd round message from P'_2 (and therefore from the simulator acting as P'_2). We can conclude that this pattern can be emulated by $\mathcal{A}'_{\mathsf{SA}}$ while interacting with $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$.
- **Pattern 3:** $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ rewinds the second round $P2P$ round for P'_{out}. Because the adversary behaves (and therefore parties P'_2 and P'_{out}) in a non-rushing way in all rounds except the first, a new second round received by P'_{out} will affect only the 4th round message sent by P'_1, however P'_2 (and therefore the simulator acting as P'_2) does not receive any message in the 4th round in Π'. We can conclude that this pattern can be emulated by $\mathcal{A}'_{\mathsf{SA}}$ while interacting with $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ for a similar reason to the one described in Theorem 12.

- **Pattern 4:** $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ rewinds the third round. This pattern is equivalent to the Pattern 4 of Theorem 17, and we follow exactly the same proof.
- **Pattern 5:** $\mathcal{S}^{\mathsf{SA}}_{1,\mathsf{out}}$ rewinds the fourth round. This pattern is equivalent to the Pattern 5 of Theorem 17, and we follow exactly the same proof.

Then with the above patterns, we show that $\mathcal{A}'_{\mathsf{SA}}$ is able to extract input from honest P'_1. By using the same ideal functionality and following the same procedures in Theorem 12, we reach a contradiction, and this result can be extended to n-party cases similar to Theorem 12. □

References

[ACGJ18] Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Round-optimal secure multiparty computation with honest majority. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 395–424, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

[ACJ17] Prabhanjan Ananth, Arka Rai Choudhuri, and Abhishek Jain. A new approach to round-optimal secure multiparty computation. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 468–499, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[BGJ+18] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Amit Sahai. Promise zero knowledge and its applications to round optimal MPC. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 459–487, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

[BHP17] Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 645–677, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 134–153, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany.

[BL18] Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 500–532, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[BMMR21] Saikrishna Badrinarayanan, Peihan Miao, Pratyay Mukherjee, and Divya Ravi. On the round complexity of fully secure solitary MPC with honest majority. *IACR Cryptol. ePrint Arch.*, page 241, 2021.

[CCG+19] Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Round optimal secure multiparty computation from minimal assumptions. Cryptology ePrint Archive, Paper 2019/216, 2019. `https://eprint.iacr.org/2019/216`.

[CCG+20] Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Round optimal secure multiparty computation from minimal assumptions. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part II*, volume 12551 of *Lecture Notes in Computer Science*, pages 291–319, Durham, NC, USA, November 16–19, 2020. Springer, Heidelberg, Germany.

[CDv88] David Chaum, Ivan Damgård, and Jeroen van de Graaf. Multiparty computations ensuring privacy of each party's input and correctness of the result. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 87–119, Santa Barbara, CA, USA, August 16–20, 1988. Springer, Heidelberg, Germany.

[CGZ20] Ran Cohen, Juan A. Garay, and Vassilis Zikas. Broadcast-optimal two-round MPC. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 828–858, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.

[Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *18th Annual ACM Symposium on Theory of Computing*, pages 364–369, Berkeley, CA, USA, May 28–30, 1986. ACM Press.

[COWZ22] Michele Ciampi, Rafail Ostrovsky, Hendrik Waldner, and Vassilis Zikas. Round-optimal and communication-efficient multiparty computation. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part I*, volume 13275 of *Lecture Notes in Computer Science*, pages 65–95, Trondheim, Norway, May 30 – June 3, 2022. Springer, Heidelberg, Germany.

[CRSW22] Michele Ciampi, Divya Ravi, Luisa Siniscalchi, and Hendrik Waldner. Round-optimal multi-party computation with identifiable abort. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part I*, volume 13275 of *Lecture Notes in Computer Science*, pages 335–364, Trondheim, Norway, May 30 – June 3, 2022. Springer, Heidelberg, Germany.

[DMR+21] Ivan Damgård, Bernardo Magri, Divya Ravi, Luisa Siniscalchi, and Sophia Yakoubov. Broadcast-optimal two round MPC with an honest majority. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 155–184, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany.

[DRSY23] Ivan Damgård, Divya Ravi, Luisa Siniscalchi, and Sophia Yakoubov. Minimizing setup in broadcast-optimal two round MPC. In *Advances in Cryptology – EUROCRYPT 2023, Part II*, Lecture Notes in Computer Science, pages 129–158. Springer, Heidelberg, Germany, June 2023.

[DS83] Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 12(4):656–666, 1983.

[FL82] Michael J. Fischer and Nancy A. Lynch. A lower bound for the time to assure interactive consistency. *Inf. Process. Lett.*, 14(4):183–186, 1982.

[GJPR21] Aarushi Goel, Abhishek Jain, Manoj Prabhakaran, and Rajeev Raghunath. On communication models and best-achievable security in two-round MPC. In Kobbi Nissim and Brent Waters, editors, *TCC 2021: 19th Theory of Cryptography Conference, Part II*, volume 13043 of *Lecture Notes in Computer Science*, pages 97–128, Raleigh, NC, USA, November 8–11, 2021. Springer, Heidelberg, Germany.

[GK96] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, 1996.

[GL05] Shafi Goldwasser and Yehuda Lindell. Secure multi-party computation without agreement. *Journal of Cryptology*, 18(3):247–287, July 2005.

[GMPP16] Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 448–476, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press.

[GS18] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 468–499, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[HHPV18] Shai Halevi, Carmit Hazay, Antigoni Polychroniadou, and Muthuramakrishnan Venkitasubramaniam. Round-optimal secure multi-party computation. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 488–520, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

[HV16] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. What security can we achieve within 4 rounds? In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16: 10th International Conference on Security in Communication Networks*, volume 9841 of *Lecture Notes in Computer Science*, pages 486–505, Amalfi, Italy, August 31 – September 2, 2016. Springer, Heidelberg, Germany.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman and Hall/CRC, 2nd edition, 2014.

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*,

volume 3152 of *Lecture Notes in Computer Science*, pages 335–354, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 735–763, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[PRS20] Arpita Patra, Divya Ravi, and Swati Singla. On the exact round complexity of best-of-both-worlds multi-party computation. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part III*, volume 12493 of *Lecture Notes in Computer Science*, pages 60–91, Daejeon, South Korea, December 7–11, 2020. Springer, Heidelberg, Germany.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press.