Broadcast-Optimal Two Round MPC with an Honest Majority

Ivan Damgård, Bernardo Magri, Luisa Siniscalchi, and Sophia Yakoubov

Aarhus University, Denmark

Abstract. This paper closes the question of the possibility of two-round MPC protocols achieving different security guarantees with and without the availability of broadcast in any given round. Cohen *et al.* [CGZ20] study this question in the dishonest majority setting; we complete the picture by studying the honest majority setting.

In the honest majority setting, given broadcast in both rounds, it is known that the strongest guarantee — guaranteed output delivery — is achievable [GLS15]. We show that in this setting, given broadcast in the first round only, guaranteed output delivery is still achievable. Given broadcast in the second round only, identifiable abort and all weaker guarantees are achievable, but fairness — and thus guaranteed output delivery — are not. Finally, using only peer-to-peer channels, for corruption thresholds $t > 1$ we show that the weakest guarantee — selective abort — is the only one achievable. For $t = 1$ and $n \geq 4$, it is known [IKP10,IKKP15] that guaranteed output delivery (and thus all weaker guarantees) are possible. We show that for $t = 1$ and $n = 3$ the strongest achievable guarantee is selective abort, definitively resolving the question of best achievable guarantees in two-round secure computation protocols.

Table of Contents

1 Introduction

In this paper we advance the study of round-optimal secure computation, focusing on secure computation with active corruptions, an honest majority, and some setup (e.g. a public key infrastructure). It is known that in this setting, secure computation is possible in two rounds. However, most known two-round protocols in the honest majority setting either only achieve the weakest security guarantee (selective abort) [ACGJ19], or make use of a broadcast channel in both rounds [GLS15].

The only exception is the protocol of Cohen *et al.* [CGZ20], which achieves secure computation with a stronger guarantee — unanimous abort — for a dishonest majority (and thus also for an honest majority) with broadcast in the second round only. Cohen *et al.* also showed that, given a dishonest majority, selective abort is the strongest achievable security guarantee with broadcast in the first round only, and unanimous abort is the strongest achievable guarantee with broadcast in the second round only.

We make a study analogous to the work of Cohen *et al.* but in the honest majority setting. Like Cohen *et al.*, we consider all four broadcast patterns: broadcast in both rounds, broadcast in the second round only, broadcast in the first round only, and no broadcast at all. Gordon *et al.* [GLS15] showed that, given broadcast in both rounds, the strongest guarantee — guaranteed output delivery — is achievable. For each of the other broadcast patterns, we ask:

> What is the strongest achievable security guarantee in this broadcast pattern, given an honest majority?

We consider the following secure computation guarantees:

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

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

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

Fairness (FAIR): A secure computation protocol achieves *fairness* if either all parties obtain the output, or none of them do. In particular, an adversary cannot learn the output if the honest parties do not also learn it.

Guaranteed Output Delivery (GOD): A secure computation protocol achieves *guaranteed output delivery* if all honest parties will learn the computation output no matter what the adversary does.

Some of these guarantees are strictly stronger than others. In particular, guaranteed output delivery implies identifiable abort (since an abort never happens), which implies unanimous abort, which in turn implies selective abort. Similarly, guaranteed output delivery implies fairness, which implies unanimous abort.

Fairness and identifiable abort are incomparable. In a fair protocol, in case of an abort, both corrupt and honest parties get less information: corrupt parties are guaranteed to learn nothing if the protocol aborts, but honest parties may not learn anything about corrupt parties' identities. On the other hand, in a protocol with identifiable abort, in case of an abort corrupt parties may learn the output, but honest parties will identify at least one corrupt party.

In Table 1, we summarize our results. Like the impossibility results of Cohen *et al.*, all of our impossibility results hold given arbitrary setup (such as a common reference string, a public key infrastructure, and correlated randomness). Our feasibility results require setups of varying complexity. Below we give a very brief description of our results. In section 1.1 we give a longer overview of our results, and the techniques we use.

No Broadcast In this setting, we show that if the adversary controls two or more parties ($t > 1$), selective abort is the best achievable guarantee. This completes the picture, since (1) selective abort can indeed be achieved by the results of Ananth *et al.* [ACGJ19], and (2) for $t = 1$, guaranteed output delivery can be achieved by the results of Ishai *et al.* [IKP10], [IKKP15].

Broadcast in the First Round Only In this setting, we show that guaranteed output delivery — the strongest guarantee — can be achieved.

Broadcast in the Second Round Only In this setting, we show that fairness is impossible if $t \geq n/3$, and also if $t > 1$. If fairness is ruled out, the best one can hope for is identifiable abort, and we show this can indeed be achieved given an honest majority.

1.1 Technical Overview

In this section we summarize our results given each of the broadcast patterns in more detail.

No Broadcast (P2P-P2P) Without a broadcast channel, we show that if an adversary controls two or more parties ($t > 1$), only the weakest guarantee — selective abort — is achievable. Ananth *et al.* [ACGJ19] give a protocol for secure computation with selective abort in this setting; we prove that secure computation with unanimous abort is not achievable.

Result 1 (Cor 1: P2P-P2P, UA, $t > 1$) *Secure computation of general functions with unanimous abort cannot be achieved in two rounds of peer-to-peer communication for corruption threshold $t > 1$.*

We prove this by focusing on broadcast, where only one party (the dealer) has an input bit, and all parties should output that bit. Specifically, we show that computing broadcast with unanimous abort in two peer-to-peer rounds with $t > 1$ is impossible[1].

[1] It is well known that computing broadcast with guaranteed output delivery requires t rounds, but this of course does not imply the same for broadcast with unanimous abort.

Broadcast Pattern R1	Broadcast Pattern R2	t	selective abort	unanimous abort	identifiable abort	fairness	guaranteed output delivery
full	full	$\frac{n}{2} \leq t < n$	✓ ⟵	✓[GS18,BL18]	← ✓[CGZ20]	—	—
p2p	full	$\frac{n}{2} \leq t < n$	✓ ⟵	✓[CGZ20]	✗[CGZ20]	—	—
full	p2p	$\frac{n}{2} \leq t < n$	✓ ↑	✗[CGZ20] ⟶ ↓	✗ ↓	—	—
p2p	p2p	$\frac{n}{2} \leq t < n$	✓[CGZ20]	✗	✗	—	—
full	full	$\frac{n}{3} \leq t < \frac{n}{2}$	✓(Obs 2)	✓(Obs 2)	✓(Obs 2)	✓ ⟵	✓[GLS15]
p2p	full	$\frac{n}{3} \leq t < \frac{n}{2}$	✓(Obs 2)	✓(Obs 2)	✓? (Conj 1)	✗(Cor 3) ⟶	✗
full	p2p	$\frac{n}{3} \leq t < \frac{n}{2}$	✓(Obs 2)	✓ ⟵	✓(Theorem 7)	✓ ⟵	✓(Theorem 7)
p2p	p2p	$\frac{n}{3} \leq t < \frac{n}{2}$	✓[ACGJ19]	✗(Cor 1) for $t > 1$ ⟶	✗ for $t > 1$	✗(Cor 1) for $t > 1$ ⟶	✗ for $t > 1$
full	full	$t < \frac{n}{3}$	✓(Obs 2)	✓(Obs 2)	✓(Obs 2)	✓(Obs 2)	✓(Obs 2)
p2p	full	$t < \frac{n}{3}$	✓(Obs 2)	✓(Obs 2)	✓(Theorem 8)	✗(Cor 4) for $t > 2$ ⟶	✗ for $t > 2$
full	p2p	$t < \frac{n}{3}$	✓(Obs 2)	✓ ⟵	✓(Theorem 7)	✓ ⟵	✓(Theorem 7)
p2p	p2p	$t < \frac{n}{3}$	✓(Obs 2)	✗(Cor 1) for $t > 1$ ⟶	✗ for $t > 1$	✗(Cor 1) for $t > 1$ ⟶	✗ for $t > 1$
p2p	p2p	$t = 1,\ n = 2$	✓(Obs 2) ⟶	✓ ⟶	✓ (Obs 1)	✗ [Cle86] ⟶	✗
p2p	p2p	$t = 1,\ n = 3$	✓(Obs 2)	✗(Cor 2) ⟶	✗	✗ (Cor 2) ⟶	✗
p2p	p2p	$t = 1,\ n = 4$	✓ ⟵	✓ ⟵	✓ ([IKKP15])	✓ ⟵	✓ ([IKKP15])
p2p	p2p	$t = 1,\ n \geq 5$	✓ ⟵	✓ ⟵	✓ ([IKP10])	✓ ⟵	✓ ([IKP10])

Table 1: Feasibility and impossibility for two-round MPC with different guarantees and broadcast patterns. (The R1 column describes whether broadcast is available in round 1; the R2 column describes whether broadcast is available in round 2.) Arrows indicate implication: the possibility of a stronger security guarantee implies the possibility of weaker ones in the same setting, and the impossibility of a weaker guarantee implies the impossibility of stronger ones in the same setting.

If an adversary controls only one party ($t = 1$), in Section 3.2 we show that for $n = 3$, selective abort is still the strongest achievable guarantee. (Patra and Ravi [PR18] give a similar result in the absence of a PKI and correlated randomness; our impossibility result is stronger, as it holds even given arbitrary correlated randomness.)

Result 2 (Cor 2: P2P-P2P, UA, $t = 1$, $n = 3$) *Secure computation of general functions with unanimous abort cannot be achieved in two rounds of peer-to-peer communication for corruption threshold $t = 1$ when $n = 3$.*

For $n \geq 5$ and $t = 1$, Ishai *et al.* [IKP10] show that the strongest guarantee — guaranteed output delivery — is achievable in two rounds of peer-to-peer communication. (A different) Ishai *et al.* [IKKP15] then show the same for $n = 4$. So, our impossibility result for $n = 3$ completes the picture in the case of one corrupt party. (Note that, for $n = 2$ and $t = 1$, we are no longer in an honest majority setting, so fairness is known to be impossible [Cle86]. Selective abort in this setting is possible, and equivalent to both unanimous and identifiable abort, since an aborting honest party is always in agreement with itself, and can always identify its only peer as the cheater.)

Broadcast in the First Round Only (BC-P2P) We use the techniques of Cohen *et al.* [CGZ20] to compile a protocol which achieves guaranteed output delivery given two rounds of broadcast (as in Gordon *et al.* [GLS15]) into a protocol using broadcast only in the first round.

Result 3 (Theorem 7: BC-P2P, GOD, $n \geq 2t + 1$) *Secure computation of general functions with guaranteed output delivery is possible in two rounds of communication, only the first of which is over a broadcast channel, for corruption threshold t such that $n \geq 2t + 1$.*

The approach of Cohen *et al.* is to have each party leverage the available round of broadcast to reliably communicate a garbled circuit corresponding to the code they use to compute their second-round message (given their input and all the first-round messages they receive). Using broadcast in the first round, parties can also reliably communicate their first-round messages. The remaining challenge is communicating the labels for the garbled circuits corresponding to those first-round messages. Cohen *et al.* use additive secret sharing; in the first round, parties additively share the labels (for their own garbled circuit), and in the second round, they forward the shares of labels (for all garbled circuits) corresponding to the first-round messages they received. In order to achieve guaranteed output delivery rather than unanimous abort, we use (verifiable) threshold secret sharing instead of additive secret sharing. We also leverage non-interactive zero knowledge proofs to enforce consistent garbling and secret sharing. We give the details of this construction in Section 4.2.

Broadcast in the Second Round Only (P2P-BC) When broadcast is available in the second round, not the first, it becomes more challenging to achieve fairness (which is implied by guaranteed output delivery). In Section 3.2, we show that fairness cannot be achieved with $n \leq 3t$.

Result 4 (Cor 3: P2P-BC, FAIR, $n \leq 3t$) *Secure computation of general functions with fairness cannot be achieved in two rounds of communication, only the second of which is over a broadcast channel, for corruption threshold t such that $n \leq 3t$.*

We show this by arguing that if the protocol is fair and $n \leq 3t$, then a corrupt player can send inconsistent messages in the first round and then use the second round messages to obtain two different outputs, corresponding to different choices of her own input − which, of course, violates privacy.

In Section 3.3, we use similar techniques to argue that fairness cannot be achieved when $t > 2$ in this setting.

Result 5 (Cor 4: P2P-BC, FAIR, $t > 2$) *Secure computation of general functions with fairness cannot be achieved in two rounds of communication, only the second of which is over a broadcast channel, for corruption threshold $t > 2$.*

For $n > 3t$, in Section 4.3 we show how to achieve identifiable abort.

Result 6 (Theorem 8: P2P-BC, ID, $n > 3t$) *Secure computation of general functions with identifiable abort is achievable in two rounds of communication, only the second of which is over a broadcast channel, for corruption threshold t such that $n > 3t$.*

Being limited to peer-to-peer channels in the first round is a more difficult setting. The main challenge is that a party P_i may send an incorrect message, or nothing at all, to party P_j in the first round. This may prevent P_j from doing what she is supposed to in the second round. When P_j complains in the second round it becomes clear that either P_i or P_j is corrupt, but since we want identifiable abort, we must either find out who to blame or compute the correct output anyway, without any further interaction.

We solve this by combining techniques similar to those of Cohen *et al.* with a new trick, where P_j will broadcast a key shared with P_i in case something was wrong in the first round. Then, when P_i broadcasts her second round message msg_i^2, we make sure that either this message is incorrect, so we can blame P_i, or the shared key from P_j allows honest parties to decrypt enough of the information in msg_i^2 to complete the computation.

Finally, we argue that the approach we use for $n > 3t$ cannot extend to $n = 3t$, but that we can leverage stronger assumptions (such as correlated randomness and obfuscation) in that setting. We sketch a protocol achieving secure computation of general functions with identifiable abort using correlated randomness and obfuscation for corruption threshold t such that $n > 2t$.

1.2 Open Questions

There are a number of questions still open. First, given a broadcast only in the second round, it is unknown whether it is possible to get fairness or guaranteed output delivery with $t = 1$ and $n = 3$, and $t = 2$ and $n \geq 7$.

Additionally, in that same setting (given a broadcast only in the second round), getting a construction achieving identifiable abort with $\frac{n}{3} \leq t < \frac{n}{2}$ without using correlated randomness or relying on obfuscation is an open problem (Conj 1 refers to a sketched protocol achieving identifiable abort in that setting with these flaws. Theorem 8 gives a construction without these flaws, but with $t < \frac{n}{3}$).

1.3 Related Work

The quest for optimal round-complexity for secure computation protocols is a well-established topic in cryptography. Starting with the first feasibility results from almost 35 years ago [Yao86,GMW87,BGW88,CCD88] a lot of progress has been made in improving the round complexity of protocols [GIKR01,Lin01,CD01] [IK02,IKP10,IKKP15,GLS15,PR18,ACGJ18,CGZ20]. In this section we detail the prior work that specifically targets the two-round setting. We divide the discussion into two: impossibility and feasibility results.

Impossibility Results. Table 2 summarizes the known lower bounds on two-round secure computation. Gennaro *et al.* [GIKR02] shed light on the optimal round-complexity for general MPC protocols achieving fairness without correlated randomness (e.g., PKI). Their model allows for communication over both authenticated point-to-point channels and a broadcast channel. They show that in this setting, three rounds are necessary for a protocol with at least $t \geq 2$ corrupt parties by focusing on the computation of exclusive-or and conjunction functions. In a slightly different model, where the parties can communicate only over a broadcast channel, Gordon *et al.* [GLS15] show that the existence of a fair two-round MPC protocol for an honest majority would imply a virtual black-box program obfuscation scheme, which would contradict the well-known impossibility result of Barak *et al.* [BGI$^+$01]. Patra and Ravi [PR18] investigate the three party setting. They show that three rounds are necessary for generic secure computation achieving unanimous abort when parties *do not* have access to a broadcast channel, and that the same three are necessary for fairness even when parties do have a broadcast channel.

It is well known that in the dishonest majority setting fairness cannot be achieved for generic computation [Cle86]. Cohen *et al.* [CGZ20] study the feasibility of two round secure computation with unanimous and identifiable abort in the dishonest majority setting. Their results show that considering arbitrary setup (e.g., a PKI) and communication over point-to-point channels, achieving unanimous abort in two rounds is not possible even if the parties are additionally allowed to communicate over a broadcast channel only in the first round, and achieving identifiable abort in two rounds is not possible even if the parties

Result	n	t	Guarantee	CRS?	PKI?	CR?	R1	R2
[GIKR02]	any	$t \geq 2$	fairness	✓	✗	✗	BC + P2P	BC + P2P
[GLS15]	any	$t < \frac{n}{2}$	fairness	✓	✗	✗	BC	BC
[PR18]	$n = 3$	$t = 1$	fairness	✓	✗	✗	BC + P2P	BC + P2P
[PR18]	$n = 3$	$t = 1$	UA	✓	✗	✗	P2P	P2P
[CGZ20]	$n = 3$	$t = 2$	UA	✓	✓	✓	BC	P2P
[CGZ20]	$n = 3$	$t = 2$	IA	✓	✓	✓	P2P	BC

Table 2: Previous impossibility results. Each row in this table describes a setting where MPC is known to be *impossible.* "UA" stand for unanimous abort, and "IA" for identifiable abort.

are additionally allowed to communicate over a broadcast channel only in the second round.

Feasibility Results. Table 3 summarizes known two-round secure computation constructions. While three rounds are necessary for fair MPC [GIKR02] for $t \geq 2$ (without correlated randomness), Ishai *et al.* [IKP10] show that it is possible to build generic two-round MPC with guaranteed output delivery when only a *single* party is corrupt ($t = 1$) for $n \geq 5$. Later, [IKKP15] showed the same for $n = 4$, and that selective abort is also possible for $n = 3$.

The work of [GLS15] gives a three round generic MPC protocol that guarantees output delivery and is secure against a minority of semi-honest fail-stop adversaries where parties only communicate over point-to-point channels; the same protocol can be upgraded to be secure against malicious adversaries if the parties are also allowed to communicate over a broadcast channel. Moreover, assuming a PKI, the protocol can be compressed to only two rounds.

For $n = 3$ and $t = 1$, Patra and Ravi [PR18] present a tight upper bound achieving unanimous abort in the setting with point-to-point channels and a broadcast channel. The protocol leverages garbled circuits, (equivocal) non-interactive commitment scheme and a PRG. In the same honest majority setting but for arbitrary n, Ananth *et al.* [ACGJ18] build four variants of a two-round protocol. Two of these variants are in the plain model (without setup), with both point-to-point channels and broadcast available in both rounds. The first achieves security with unanimous abort and relies on one-way functions, and the second achieves guaranteed output delivery against fail-stop adversaries and additionally relies on semi-honest oblivious transfer. Their other two protocols require a PKI; and achieve guaranteed output delivery against fail-stop and semi-malicious adversaries.

Finally, Cohen *et al.* [CGZ20] present a complete characterization of the feasibility landscape of two-round MPC in the dishonest majority setting, for all broadcast patterns. In particular, they show protocols (without setup) for the cases of point-to-point communication in both rounds, point-to-point in the first

Result	n	t	Guarantee	PKI?	CRS?	1st round	2nd round	Assumptions
[IKP10]	$n \geq 5$	$t = 1$	GOD	✗	✗	P2P	P2P	PRG
[IKKP15]	$n = 3$	$t = 1$	SA	✗	✗	P2P	P2P	PRG
[IKKP15]	$n = 4$	$t = 1$	GOD	✗	✗	P2P	P2P	injective OWF
[GLS15]	any	$t < \frac{n}{2}$	M-GOD	✓	✓	BC + P2P	BC + P2P	dFHE
[PR18]	$n = 3$	$t = 1$	UA	✓	✓	BC + P2P	BC + P2P	GC, NICOM, eNICOM, PRG
[ACGJ18]	any	$t < \frac{n}{2}$	UA	✗	✗	BC + P2P	BC + P2P	OWF
[ACGJ18]	any	$t < \frac{n}{2}$	FS-GOD	✓	✗	BC + P2P	BC + P2P	OWF
[ACGJ18]	any	$t < \frac{n}{2}$	FS-GOD	✗	✗	BC + P2P	BC + P2P	OWF, SH-OT
[ACGJ18]	any	$t < \frac{n}{2}$	FS-GOD / SM-GOD	✓	✗	BC	BC	OWF
[CGZ20]	any	$t < n$	SA	✗	✓	P2P	P2P	2-round OT
[CGZ20]	any	$t < n$	UA	✗	✓	P2P	BC	2-round OT
[CGZ20]	any	$t < n$	IA	✗	✓	BC	BC	2-round OT

Table 3: Protocols for secure MPC with two-rounds. "UA" stands for unanimous abort, "FS-GOD" for guaranteed output delivery against fail-stop adversaries, "SM-GOD" for guaranteed output delivery against semi-malicious adversaries, and "M-GOD" for guaranteed output delivery against malicious adversaries.

round and broadcast in the second round, and broadcast in both rounds. The protocols achieve security with selective abort, unanimous abort and indentifiable abort, respectively. All protocols rely on two-round oblivious transfer.

2 Secure Multiparty Computation (MPC) Definitions

2.1 Security Model

We follow the real/ideal world simulation paradigm and we adopt the security model of Cohen, Garay and Zikas [CGZ20]. As in their work, we state our results in a stand-alone setting.[2]

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 corrupted parties, their respective private inputs, and an auxiliary input. The protocol is executed in rounds (i.e., the protocol is synchronous), where 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 point-to-point (P2P) network, where we additionally assume all communication to be private and ideally authenticated. (Given a PKI and a broadcast channel, such a fully connected point-to-point network can be instantiated.)

During the execution of the protocol, the corrupt parties receive arbitrary instructions from the adversary $\mathcal{A}$, while the honest parties faithfully follow the

[2] We note that our security proofs can translate to an appropriate (synchronous) composable setting with minimal changes.

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, the corrupt parties produce no output, and the adversary outputs an arbitrary function of its view. The view of a party during the execution consists of its input, random coins and the messages it sees during the execution.

Definition 1 (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 security parameter λ, denoted $\mathtt{REAL}_{\Pi,\mathcal{I},\mathcal{A}(\mathsf{aux})}(x, \lambda)$, 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 ($\mathsf{sl\text{-}abort}$), unanimous abort ($\mathsf{un\text{-}abort}$), identifiable abort ($\mathsf{id\text{-}abort}$), fairness ($\mathsf{fairness}$) and guaranteed output delivery (god).

Definition 2 (Ideal Computation). *Consider* $\mathsf{type} \in \{\mathsf{sl\text{-}abort}, \mathsf{un\text{-}abort}, \mathsf{id\text{-}abort}, \mathsf{fairness}, \mathsf{god}\}$. *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 corrupted 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 security parameter λ, denoted $\mathtt{IDEAL}^{\mathsf{type}}_{f,\mathcal{I},\mathcal{S}(\mathsf{aux})}(x, \lambda)$, 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 adversary may send to the trusted party arbitrary inputs for the corrupted parties. Let x'_i be the value actually sent as the input of party P_i.*
2. Trusted party answers adversary*: The trusted party computes $y = f(x'_1, \ldots, x'_n)$. If there are no corrupted parties or* $\mathsf{type} = \mathsf{god}$, *proceed to step 4.*
   (a) If $\mathsf{type} \in \{\mathsf{sl\text{-}abort}, \mathsf{un\text{-}abort}, \mathsf{id\text{-}abort}\}$: *The trusted party sends y to $\mathcal{S}$.*
   (b) If $\mathsf{type} = \mathsf{fairness}$: *The trusted party sends* ready *to $\mathcal{S}$.*
3. Adversary $\mathcal{S}$ responds to trusted party*:*
   (a) If $\mathsf{type} = \mathsf{sl\text{-}abort}$: *The adversary $\mathcal{S}$ 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 $\mathsf{type} \in \{\mathsf{un\text{-}abort}, \mathsf{fairness}\}$: *The adversary can send* $\mathtt{abort}$ *to the trusted party. If it does, we take $\mathcal{J} = [n] \setminus \mathcal{I}$.*
   (c) If $\mathsf{type} = \mathsf{id\text{-}abort}$: *If it chooses to abort, the adversary $\mathcal{S}$ 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 adversary $\mathcal{S}$,*
      i. It sets the abort message $\mathsf{abortmsg}$, *as follows:*

– *if* $\mathsf{type} \in \{\mathsf{sl\text{-}abort}, \mathsf{un\text{-}abort}, \mathsf{fairness}\}$*, we let* $\mathsf{abortmsg} = \bot$*.*
– *if* $\mathsf{type} = \mathsf{id\text{-}abort}$*, we let* $\mathsf{abortmsg} = (\bot, i^*)$*.*

ii. The trusted party then sends $\mathsf{abortmsg}$ *to every party* P_j*,* $j \in \mathcal{J}$*, and* y *to every party* P_j*,* $j \in [n] \setminus \mathcal{J}$*.*

Note that, if $\mathsf{type} = \mathsf{god}$*, we will never be in this setting, since* $\mathcal{S}$ *was not allowed to ask for an abort.*

(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 corrupted parties output nothing, The adversary* $\mathcal{S}$ *outputs an arbitrary function of the initial inputs* $\{x_i\}_{x \in \mathcal{I}}$*, the messages received by the corrupted parties from the trusted party and its auxiliary input.*

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

Definition 3. *Consider* $\mathsf{type} \in \{\mathsf{sl\text{-}abort}, \mathsf{un\text{-}abort}, \mathsf{id\text{-}abort}, \mathsf{fairness}, \mathsf{god}\}$*. 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* $\mathcal{A}$ *there exists a PPT simulator* $\mathcal{S}$ *such that for every* $\mathcal{I} \subseteq [n]$ *of size at most* t*, it holds that*

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

2.2 Notation

In this paper, we focus on two-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 three algorithms: $\mathtt{frst\text{-}msg}_i$, to compute P_i's first messages to its peers; $\mathtt{snd\text{-}msg}_i$, to compute P_i's second messages; and $\mathtt{output}_i$, to compute P_i's output. Thus, a protocol Π can be defined as $\{(\mathtt{frst\text{-}msg}_i, \mathtt{snd\text{-}msg}_i, \mathtt{output}_i)\}_{i\in[n]}$.

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.
- $\mathtt{snd\text{-}msg}_i(x_i, r_i, \mathsf{msg}_{1,1\to i}, \ldots, \mathsf{msg}_{1,n\to i}) \to (\mathsf{msg}_{2,i\to 1}, \ldots, \mathsf{msg}_{2,i\to n})$ produces the second-round messages of party P_i to all parties.
- $\mathtt{output}_i(x_i, r_i, \mathsf{msg}_{1,1\to i}, \ldots, \mathsf{msg}_{1,n\to i}, \mathsf{msg}_{2,1\to i}, \ldots, \mathsf{msg}_{2,n\to i}) \to y_i$ produces the output returned to party P_i.

Throughout our negative results, we omit the randomness r, and instead focus on deterministic protocols, modeling the randomness implicitly as part of the algorithm.

3 Negative Results

For some of our negative results, we leverage similar negative results for broadcast (or byzantine agreement). To show that guaranteed output delivery is impossible in two rounds of peer-to-peer communication, we can use the fact that broadcast cannot be realized in two rounds for $t > 1$ [FL82,DS83]. To show the impossibility of weaker guarantees such as unanimous abort in this setting, we prove (in Section 3.1) that a weaker flavor of broadcast, called *(weak) detectable* broadcast [FGMv02] — where all parties either learn the broadcast bit, or unanimously abort — cannot be realized in two rounds either.

Of course, these techniques do not suffice to prove impossibility when at least one round of broadcast is available. In Sections 3.2 and 3.3, we leverage a protocol property which we call *last message resiliency* to prove several impossibility results in those settings. Informally, a protocol is last message resilient if its output function can produce the correct output given only a subset of the last round messages.

3.1 Impossibility of Weak Detectable Broadcast in Two Rounds

We state the definitions of *broadcast* and *detectable broadcast* (from Fitzi *et al.* [FGMv02]) below.

Definition 4 (Broadcast). *A protocol among n parties, where the dealer $D = P_1$ holds an input value $x \in \{0, 1\}$ and every other party $P_i, i \in [2, \ldots, n]$ outputs a value $y_i \in \{0, 1\}$, achieves* broadcast *if it satisfies the following two conditions:*

Validity: *If the dealer D is honest then all honest parties P_i output $y_i = x$.*
Consistency: *All honest parties output the same value $y_2 = \cdots = y_n = y$.*

Definition 5 (Detectable Broadcast). *A protocol among n parties achieves* detectable broadcast *if it satisfies the following three conditions:*

Correctness: *All honest parties unanimously accept or unanimously reject the protocol. If all honest parties accept then the protocol achieves broadcast.*
Completeness: *If all parties are honest then all parties accept.*
Fairness: *If any honest party rejects the protocol then the adversary gets no information about the dealer's input x.*

We additionally define *weak detectable broadcast.*

Definition 6 (Weak Detectable Broadcast). *A protocol among n parties achieves* weak detectable broadcast *if it satisfies only the correctness and completeness requirements of detectable broadcast.*

An alternative way of viewing broadcast, through the lense of secure computation, is by considering the simple broadcast function $f_{\mathsf{bc}}(x, \perp, \ldots, \perp) = (\perp, x, \ldots, x)$ which takes an input bit x from the dealer $D = P_1$, and outputs that bit to all other parties. *Broadcast* (Definition 4) is exactly equivalent to computing f_{bc} with guaranteed output delivery; *detectable broadcast* (Definition 5) is equivalent to computing it with fairness; and *weak detectable broadcast* (Definition 6) is equivalent to computing it with unanimous abort.

Theorem 1. *Weak detectable broadcast cannot be achieved in two rounds of peer-to-peer communication for corruption threshold $t > 1$.*

Proof. We prove Theorem 1 by contradiction. We let

$$\Pi_{\mathtt{wdbc}} = \{(\mathtt{frst\text{-}msg}_i, \mathtt{snd\text{-}msg}_i, \mathtt{output}_i)\}_{i\in[1,\ldots,n]}$$

be the description of the two-round weak detectable broadcast protocol. (We use the notation we introduce for two-round secure computation in Section 2.2, and consider the weak detectable broadcast protocol to be a secure computation with unanimous abort of f_{bc}. We let $x_1 = x$ denote the bit being broadcast by the dealer $D = P_1$, and $x_i = \perp$ for $i \in [2,\ldots,n]$ be placeholders for other parties' inputs.)

Below we describe a sequence of adversarial strategies with two corruptions ($t = 2$). The dealer $D = P_1$ is corrupt in all of these; at most one of the receiving parties $P_2,\ldots,P_n$ is corrupt at a time. Each subsequent strategy clearly requires certain parties to output certain values, by the definition of weak detectable broadcast. In the last strategy, we see a contradiction, where one party must output both 0 and 1. Therefore, $\Pi_{\mathtt{wdbc}}$ could not have been a weak detectable broadcast protocol.
In all of the strategies below, we let $\mathsf{msg}_{b,i\to j}$ denote a party P_i's bth-round message to party P_j; we only specify how these messages are generated when this is done dishonestly.

Strategy 1:
Behavior: All parties behave honestly. D executes the protocol with $x = 0$.
Output: By completeness, all honest parties must accept the protocol. By correctness, the protocol must thus achieve broadcast. By validity, all honest parties must output 0.
Strategy 2_M:
Behavior: D and P_2 behave maliciously. D computes two different sets of first-round messages, using different inputs $x = 0$ and $x = 1$, as follows:

$$(\mathsf{msg}^{(0)}_{1,1\to1},\ldots,\mathsf{msg}^{(0)}_{1,1\to n}) \leftarrow \mathtt{frst\text{-}msg}_1(x = 0)$$

$$(\mathsf{msg}^{(1)}_{1,1\to1},\ldots,\mathsf{msg}^{(1)}_{1,1\to n}) \leftarrow \mathtt{frst\text{-}msg}_1(x = 1)$$

D sends $\mathsf{msg}^{(0)}_{1,1\to3},\ldots,\mathsf{msg}^{(0)}_{1,1\to n}$ to parties $P_3,\ldots,P_n$.
Party P_2 then computes two different sets of second-round messages, as follows:

$$(\mathsf{msg}^{(0)}_{2,2\to1},\ldots,\mathsf{msg}^{(0)}_{2,2\to n}) \leftarrow \mathtt{snd\text{-}msg}_2(\perp, \mathsf{msg}^{(0)}_{1,1\to2}, \mathsf{msg}_{1,2\to2},\ldots,\mathsf{msg}_{1,n\to2})$$

$$(\mathsf{msg}^{(1)}_{2,2\to1},\ldots,\mathsf{msg}^{(1)}_{2,2\to n}) \leftarrow \mathtt{snd\text{-}msg}_2(\perp, \mathsf{msg}^{(1)}_{1,1\to2}, \mathsf{msg}_{1,2\to2},\ldots,\mathsf{msg}_{1,n\to2})$$

P_2 sends $\mathsf{msg}^{(1)}_{2,2\to n}$ to P_n (pretending, essentially, that D dealt a 1), and $\mathsf{msg}^{(0)}_{2,2\to i}$ to other parties P_i (pretending that D dealt a 0). In the second round, all parties compute their messages honestly; D uses $x = 0$.

Output: $P_3, \ldots, P_{n-1}$ must accept and output 0, since their views are identical to those in the previous game. By correctness, P_n must also accept. By consistency, P_n must also output 0.
Strategy 2_H:
Behavior: P_2 is honest again; D is still malicious. D sends $\mathsf{msg}^{(1)}_{1,1\to 2}$ to P_2, and $\mathsf{msg}^{(0)}_{1,1\to i}$ to other parties P_i. In the second round, D continues to represent 1 to P_2 and 0 to the others.
Output: P_n must accept and output 0, since its view is the same as in the previous game. By correctness, $P_2, \ldots, P_{n-1}$ must also accept. By consistency, $P_2, \ldots, P_{n-1}$ must also output 0.

Now, skipping ahead, we generalize, for $k \in [3, \ldots, n-1]$:

Strategy k_M:
Behavior: D and P_k behave maliciously. D sends $\mathsf{msg}^{(1)}_{1,1\to i}$ to $P_2, \ldots, P_{k-1}$, and $\mathsf{msg}^{(0)}_{1,1\to i}$ to the other parties $P_{k+1}, \ldots, P_n$. In the second round, D continues to represent 1 to $P_2, \ldots, P_{k-1}$ and 0 to $P_{k+1}, \ldots, P_n$.
In the second round, as P_2 did in strategy 2_M, P_k uses $\mathsf{msg}^{(0)}_{1,1\to k}$ to compute $(\mathsf{msg}^{(0)}_{2,k\to 1}, \ldots, \mathsf{msg}^{(0)}_{2,k\to n-1})$ (which it sends to $P_2, \ldots, P_{n-1}$), and $\mathsf{msg}^{(1)}_{1,1\to k}$ to compute $\mathsf{msg}^{(1)}_{2,k\to n}$ (which it sends to P_n).
Output: $P_2, \ldots, P_{n-1}$ must accept and output 0, since their views are identical to those in the previous game. By correctness, P_n must also accept. By consistency, P_n must also output 0.
Strategy k_H:
Behavior: P_k is honest again. D sends $\mathsf{msg}^{(1)}_{1,1\to i}$ to $P_2, \ldots, P_k$, and $\mathsf{msg}^{(0)}_{1,1\to i}$ to the other parties $P_{k+1}, \ldots, P_n$. In the second round, D continues to represent 1 to $P_2, \ldots, P_k$ and 0 to $P_{k+1}, \ldots, P_n$.
Output: P_n must accept and output 0, since its view is the same as in the previous game. By correctness, $P_2, \ldots, P_{n-1}$ must also accept. By consistency, $P_2, \ldots, P_{n-1}$ must also output 0.

We end with strategies n_M, n_H.

Strategy n_M:
Behavior: D behaves honestly and executes the protocol with $x = 1$.
In the second round, P_n behaves maliciously; it pretends it got 0 towards, e.g., only P_2. More precisely, P_n uses $\mathsf{msg}^{(0)}_{1,1\to n}$ to compute $\mathsf{msg}^{(0)}_{2,n\to 2}$ (which it sends to P_2), and $\mathsf{msg}^{(1)}_{1,1\to n}$ to compute $(\mathsf{msg}^{(1)}_{2,n\to 3}, \ldots, \mathsf{msg}_{2,n\to n-2})$ (which it sends to $P_3, \ldots, P_{n-1}$).
Output: P_2 must accept and output 0, since its view is the same as in the previous game. By correctness, $P_3, \ldots, P_{n-1}$ must also accept. By consistency, $P_3, \ldots, P_{n-1}$ must also output 0.
Strategy n_H:
Behavior:
All parties behave honestly. D executes the protocol with $x = 1$.

In strategy n_H, on the one hand, by completeness, all honest parties must accept the protocol; by validity, all honest parties must output 1. On the other hand, since the view of P_2 is the same as its view in the previous strategy, P_2 must output 0! This is a contradiction.

The impossibility of realizing weak detectable broadcast in two rounds for $t > 1$ clearly implies that there exists a function (specifically, f_{bc}) which is impossible to compute with unanimous abort for $t > 1$ in two rounds of peer-to-peer communication.

Corollary 1 (P2P-P2P, UA, $t > 1$). *Secure computation of general functions with unanimous abort cannot be achieved in two rounds of peer-to-peer communication for corruption threshold $t > 1$.*

3.2 Impossibility Results for MPC with Unanimous Abort and Fairness with $n = 2t + 1$

In our impossibility results in this section, we use a property which we call *last message resiliency.*

Definition 7 (Last Message Resiliency). *A protocol is t-*last message resilient *if, in an honest execution, any protocol participant P_i can compute its output without using t of the messages it received in the last round.*

More formally, consider a protocol $\Pi = \{(\texttt{frst-msg}_i, \texttt{snd-msg}_i, \texttt{output}_i)\}_{i\in[1,\ldots,n]}$. The protocol is t-last message resilient if, for each $i \in [1, \ldots, n]$ and each $S \subseteq \{1, \ldots, n\} \backslash \{i\}$ such that $|S| \leq t$, the output function $\texttt{output}_i$ returns the correct output even without second round messages from parties $P_i, i \in S$. That is, for all security parameters λ, for all sets $S \subseteq \{1, \ldots, n\} \backslash \{i\}$ such that $|S| \leq t$, for all inputs $x_1, \ldots, x_n$,

$$\Pr\begin{pmatrix} \texttt{output}_i(x_i, \mathsf{msg}_{1,1\to i}, \ldots, \mathsf{msg}_{1,n\to i}, \mathsf{msg}'_{2,1\to i}, \ldots, \mathsf{msg}'_{2,n\to i}) \\ \neq \texttt{output}_i(x_i, \mathsf{msg}_{1,1\to i}, \ldots, \mathsf{msg}_{1,n\to i}, \mathsf{msg}_{2,1\to i}, \ldots, \mathsf{msg}_{2,n\to i}) \end{pmatrix} = negl(1^\lambda)$$

over the randomness used in the protocol, where, for $j \in [1, \ldots, n]$,

$$(\mathsf{msg}_{1,j\to 1}, \ldots, \mathsf{msg}_{1,j\to n}) \leftarrow \texttt{frst-msg}_j(x_j),$$

$$(\mathsf{msg}_{2,j\to 1}, \ldots, \mathsf{msg}_{2,j\to n}) \leftarrow \texttt{snd-msg}_j(x_j, \mathsf{msg}_{1,1\to j}, \ldots, \mathsf{msg}_{1,n\to j}),$$

and

$$\mathsf{msg}'_{2,j\to i} = \begin{cases} \mathsf{msg}_{2,j\to i}, & \textit{if } j \notin S, \\ \perp & \textit{otherwise.} \end{cases}$$

Theorem 2. *Any protocol Π which achieves secure computation with unanimous abort with corruption threshold t and whose last round can be executed over peer-to-peer channels must be t-last message resilient.*

Proof. We prove this by contradiction. Assume Π achieves unanimous abort, and is not t-resilient. Then, by definition, there exist inputs $x_1, \ldots, x_n$, an $i \in [1, \ldots, n]$ and a subset $S \subseteq \{1, \ldots, n\} \backslash \{i\}$ (such that $|S| \leq t$) where, with non-negligible probability,

$$\begin{aligned}&\mathtt{output}_i(x_i, \mathsf{msg}_{1,1\to i}, \ldots, \mathsf{msg}_{1,n\to i}, \mathsf{msg}'_{2,1\to i}, \ldots, \mathsf{msg}'_{2,n\to i})\\ &\neq \mathtt{output}_i(x_i, \mathsf{msg}_{1,1\to i}, \ldots, \mathsf{msg}_{1,n\to i}, \mathsf{msg}_{2,1\to i}, \ldots, \mathsf{msg}_{2,n\to i})\end{aligned}$$

(where the messages are produced in the way described in Definition 7).

The adversary can use this by corrupting P_j, $j \in S$; it will behave honestly, except in the last round, where $P_j, j \in S$ will not send messages to P_i. (Note that the ability to send last round messages to some parties but not others relies on the fact that the last round is over peer-to-peer channels.) With non-negligible probability, P_i will receive an incorrect output (e.g. an abort). However, this cannot occur in a protocol with unanimous abort; all other honest parties must accept the protocol and produce the correct output (since their views are the same as in an entirely honest execution), so P_i must as well.

Theorem 3. *Any protocol Π which achieves secure computation with fairness with corruption threshold t must be t-last message resilient.*

Proof. We prove this by contradiction. Assume Π achieves fairness, and is not t-resilient. Then, by definition, there exist inputs $x_1, \ldots, x_n$, an $i \in [1, \ldots, n]$ and a subset $S \subseteq \{1, \ldots, n\} \backslash \{i\}$ (such that $|S| \leq t$) where, with non-negligible probability,

$$\begin{aligned}&\mathtt{output}_i(x_i, \mathsf{msg}_{1,1\to i}, \ldots, \mathsf{msg}_{1,n\to i}, \mathsf{msg}'_{2,1\to i}, \ldots, \mathsf{msg}'_{2,n\to i})\\ &\neq \mathtt{output}_i(x_i, \mathsf{msg}_{1,1\to i}, \ldots, \mathsf{msg}_{1,n\to i}, \mathsf{msg}_{2,1\to i}, \ldots, \mathsf{msg}_{2,n\to i}).\end{aligned}$$

(where the messages are produced in the way described in Definition 7).

A rushing adversary can use this by corrupting P_j, $j \in S$. As in the previous proof, it will behave honestly, except in the last round, where $P_j, j \in S$ will not send messages to P_i. With non-negligible probability, P_i will receive an incorrect output (e.g. an abort), while the rushing adversary will learn the output, since it will have all of the messages it would have gotten in a fully honest execution of the protocol. This violates fairness.[3]

Theorem 4. *There exists a function f such that any protocol Π securely realizing f with corruption threshold t such that $n \leq 3t$ and whose first round can be executed over peer-to-peer channels cannot be t-last message resilient.*

Proof. Consider a concrete function $f_=$, which we design to emphasize that re-computation with different inputs is a clear violation of privacy. Let each party P_i, $i \in [n]$ hold as input an index $x_i \in [0, \ldots, l]$ (for, e.g., $l = n$). $f_=$ allows all

[3] Note that while P_i does not learn the output, other honest parties might. However, even one honest party not receiving the output is a violation of fairness if the adversary learns the output.

parties to learn which parties had the same inputs. (For instance, for $n = 3$, $x_1 = x_3 = 0$ and $x_2 = 1$, all parties learn that P_1 and P_3 had the same input which was different from P_2's input; they learn nothing else about other parties' inputs.) More formally,

$$f_=(x_1, \ldots, x_n) = (M, \ldots, M),$$

where

$$M = \{x_i \overset{?}{=} x_j\}_{i,j \in [n]}.$$

Consider, without loss of generality, party P_1. The adversary should clearly be unable to recompute the function with both, e.g., $x_1 = 0$ and $x_1 = 1$ (assuming the other corrupt parties' inputs are not 0 or 1). If it does, it will learn both the identities of all parties whose inputs were 0, and the identities of all parties whose inputs were 1, whereas in an ideal execution, it can learn exactly one of those two things.

We now show that, in a t-last message resilient (where $n \leq 3t$) two-round protocol Π where the first round is over peer-to-peer channels, P_1 can always learn both of those outputs. Consider a corrupt P_1, and partition the honest parties into two sets of equal size (assuming for simplicity that the number of honest parties is even): S_0 and S_1. Note that $|S_0| = |S_1| = \frac{n-t}{2} \leq t$.

P_1 uses $x_1 = 0$ to compute its first round messages to S_0; it uses $x_1 = 1$ to compute its first round messages to S_1. (Note that the ability to send first round messages based on different inputs relies on the fact that the first round is over peer-to-peer channels.) All other parties behave honestly. Because the protocol Π is t-last message resilient, and because S_1 contains t or fewer parties, P_1 has enough second round messages excluding those it received from S_1 to compute its output. Note that all second round messages except for those received from S_1 are distributed exactly as in an honest execution with $x_1 = 0$; therefore, by last message resiliency, P_1 finds out which other parties had 0 as an input. Similarly, by excluding second round messages it received from S_0, P_1 finds out which other parties had 1 as an input. This is clearly a violation of privacy.

Corollary 2 (P2P-P2P, UA, $n \leq 3t$). *Secure computation of general functions with unanimous abort cannot be achieved in two rounds of peer-to-peer communication for corruption threshold t such that $n \leq 3t$.*

This corollary follows directly from Theorems 2 and 4.

Remark 1. Note that for $t > 1$, Cor 2 is subsumed by Cor 1. However, Cor 2 covers the case of $t = 1$ and $n = 3$, closing the question of unanimous abort with honest majority in two rounds of peer-to-peer communication.

Corollary 3 (P2P-BC, FAIR, $n \leq 3t$). *Secure computation of general functions with fairness cannot be achieved in two rounds the first of which is over peer-to-peer channels for corruption threshold t such that $n \leq 3t$.*

This corollary follows from Theorems 3 and 4.

3.3 Impossibility Results for MPC with Fairness with $t > 2$

We use a similar approach to show that secure computation with guaranteed output delivery is impossible to achieve in two rounds the second of which is over peer-to-peer channels if $t > 2$. To do this, we define a stronger notion of last message resiliency.

Definition 8 (Strong Last Message Resiliency). *t-Strong last message resiliency is defined as regular t-last message resiliency, but first-round messages from $P_j, j \in S$ can be maliciously generated.*

More formally, consider a protocol $\Pi = \{(\texttt{frst-msg}_i, \texttt{snd-msg}_i, \texttt{output}_i)\}_{i\in[1,\dots,n]}$. The protocol is t-last message resilient if, for each $i \in [1,\dots,n]$ and each $S \subseteq \{1,\dots,n\}\backslash\{i\}$ such that $|S| \le t$, the output function $\texttt{output}_i$ returns the same output even without second round messages from parties $P_j, j \in S$. That is, for all security parameters λ, for all sets $S \subseteq \{1,\dots,n\}\backslash\{i\}$ such that $|S| \le t$, for all PPT algorithms $\texttt{frst-msg}^*_j, j \in S$, for all inputs $x_1,\dots,x_n$,

$$\Pr\begin{pmatrix} \texttt{output}_i(x_i, \mathsf{msg}_{1,1\to i},\dots,\mathsf{msg}_{1,n\to i}, \mathsf{msg}'_{2,1\to i},\dots,\mathsf{msg}'_{2,n\to i}) \\ \ne \texttt{output}_i(x_i, \mathsf{msg}_{1,1\to i},\dots,\mathsf{msg}_{1,n\to i}, \mathsf{msg}_{2,1\to i},\dots,\mathsf{msg}_{2,n\to i}) \end{pmatrix} = negl(1^\lambda)$$

over the randomness used in the protocol, where, for $j \in [1,\dots,n]$,

$$(\mathsf{msg}_{1,j\to 1},\dots,\mathsf{msg}_{1,j\to n}) \leftarrow \begin{cases} \texttt{frst-msg}_j(x_j) & \text{if } j \in \{1,\dots,n\}\backslash S, \\ \texttt{frst-msg}^*_j(x_j) & \text{if } j \in S, \end{cases}$$

$$(\mathsf{msg}_{2,j\to 1},\dots,\mathsf{msg}_{2,j\to n}) \leftarrow \texttt{snd-msg}_j(x_j, \mathsf{msg}_{1,1\to j},\dots,\mathsf{msg}_{1,n\to j}),$$

and

$$\mathsf{msg}'_{2,j\to i} = \begin{cases} \mathsf{msg}_{2,j\to i}, & \text{if } j \in \{1,\dots,n\}\backslash S, \\ \bot & \text{if } j \in S. \end{cases}$$

Note that, if only t' parties cheated in the first round of a t-strong last message resilient protocol (for $t' < t$), omitting *any* additional $t - t'$ second round messages should not affect the computed output.

Theorem 5. *There exists a function f such that any protocol Π securely realizing f with fairness with corruption threshold t must be $(t-1)$-strong last message resilient.*

Proof. Consider the function $f_=$ from the previous section, but augmented so that every party has a length-n *vector* of values as input, and the parties only learn whether x_i and x_j agree at indices i and j. We call the augmented function $f_{='}$. This is meant to ensure that any pair of distinct adversarial inputs reveals different information about honest parties' inputs. (Previously, if we had corrupt P_a and $P_{a'}$, $x_a = 0, x_{a'} = 1$ revealed the same information to the adversary as $x_a = 1, x_{a'} = 0$.)

We prove the theorem by contradiction. Assume Π computes $f_{='}$ with fairness, and is not $(t-1)$-strong last message resilient. Then, by definition, there exist inputs $x_1, \ldots, x_n$, an $i \in [1, \ldots, n]$ and a subset $S \subseteq \{1, \ldots, n\} \backslash \{i\}$ (such that $|S| \leq t - 1$) such that, with non-negligible probability,

$$y \neq y'$$

where

$$y = \mathtt{output}_i(x_i, \mathsf{msg}_{1,1\to i}, \ldots, \mathsf{msg}_{1,n\to i}, \mathsf{msg}'_{2,1\to i}, \ldots, \mathsf{msg}'_{2,n\to i}),$$

$$y' = \mathtt{output}_i(x_i, \mathsf{msg}_{1,1\to i}, \ldots, \mathsf{msg}_{1,n\to i}, \mathsf{msg}_{2,1\to i}, \ldots, \mathsf{msg}_{2,n\to i}),$$

and the messages are produced in the way described in Definition 8.

Recall that by the definition of secure computation with fairness, the honest output functions $\mathtt{output}_i$ will only produce the actual output on some set of inputs, or `abort`. Recall also that $f_{='}$ gives the same output to all parties. So, in a fair computation of $f_{='}$, all honest parties will output the same thing.

- If either y or y' are `abort`, the adversary can violate fairness. Without loss of generality, say $y' =$ `abort`. The adversary corrupts all parties $P_j, j \in S$. It then generates its messages as in Definition 8 and withholds the appropriate second-round messages, causing all parties to output $y' =$ `abort`. By corrupting one additional party (but allowing it to behave honestly), the adversary learns the output: it knows the second-round messages it withheld from the honest parties, and can use those messages and the additional corrupt party to compute y.
- If $y \neq y'$ and neither of them is `abort`, the adversary can violate privacy. As before, the adversary corrupts all parties $P_j, j \in S$, generates its messages as in Definition 8, and withholds the appropriate second-round messages, causing all parties to output y'. By corrupting one additional party (but allowing it to behave honestly), the adversary also learns y. y and y' must be different outputs of $f_{='}$ on the same honest party inputs; so, they must be outputs of $f_{='}$ on different adversarial inputs. Thus, for at least one corrupt party P_a, the adversary gets to test equality with the ath value in parties' input vectors *twice* instead of once, which is clearly more than an adversary would learn by interacting with a trusted third party. (Note that this is where we used the augmented $f_{='}$: for $f_{=}$, computation with different adversarial inputs does not guarantee that the adversary learns more about honest party inputs. For $f_{='}$, it does.)

Theorem 6. *There exists a function f such that any protocol Π securely realizing f with corruption threshold $t > 1$ and whose first round can be executed over peer-to-peer channels cannot be 2-strong last message resilient.*

Proof. Consider the function $f_{='}$ from the previous section. We let $y^{(0)}$ be the output on $x_1 = 0^n$, and $y^{(1)}$ be the output on $x_1 = 1^n$ (with all other inputs fixed).

We show that, in a 2-strong last message resilient two-round protocol Π where the first round is over peer-to-peer channels, P_1 and P_2 can always learn both $y^{(0)}$ and $y^{(1)}$. Below we describe a sequence of adversarial strategies with two corruptions ($t = 2$); parties P_1 and P_2 are corrupt in all of the strategies. P_1 misbehaves by sending first-round messages generated with $x_1 = 0^n$ to some parties, and first-round messages generated with $x_1 = 1^n$ to others. Though P_2 computes its messages honestly, the adversary uses its view to compute additional information.

The numbering of the strategies described below is a bit counterintuitive; we start with strategy 2, which is simply an honest execution with $x_1 = 0^n$, so P_2 should clearly learn $y^{(0)}$. After that, strategies $i \in [3, \ldots, n]$ are all strategies where $P_3, \ldots, P_i$ receive a first-round message based on $x_1 = 1^n$, and $P_{i+1}, \ldots, P_n$ receive a first-round message based on $x_1 = 0^n$.

Strategy i for $i \in [2, \ldots, n]$:
Behavior: P_1 and P_2 are corrupt. P_2 behaves honestly, but P_1 uses $x_1 = 1^n$ to compute her first round messages to $P_3, \ldots, P_i$, and $x_1 = 0^n$ to compute her first round messages to $P_{i+1}, \ldots, P_n$. (We don't care how second-round messages to $P_3, \ldots, P_n$ are computed, since we focus on the output of P_2.) The first-round messages to honest parties in Strategy 2 are identically distributed to those in an honest execution with $x_1 = 0^n$, and the first-round messages in Strategy n are identically distributed to those in an honest execution with $x_1 = 1^n$.
Defined values: For $b \in \{0, 1\}$, let $\mathsf{msg}^{(b)}_{1,1\to 2}$ and $\mathsf{msg}^{(b)}_{2,1\to 2}$ be P_1's messages to P_2 computed with $x_1 = b^n$; let $\mathsf{msg}^{(b)}_{2,2\to 2}$ be P_2's state (message to itself) given that it received $\mathsf{msg}^{(b)}_{2,1\to 2}$. All messages received from honest parties are computed by those parties honestly given the strategy i behavior of P_1. For $b \in \{0, 1\}$, we let

$$\begin{aligned} y_{b,i} = \mathtt{output}_2(x_2, &\mathsf{msg}^{(b)}_{1,1\to 2}, \mathsf{msg}_{1,2\to 2}, \mathsf{msg}_{2,3\to 2}, \ldots, \mathsf{msg}_{1,n\to 2}, \\ &\mathsf{msg}^{(b)}_{2,1\to 2}, \mathsf{msg}^{(b)}_{2,2\to 2}, \mathsf{msg}_{2,3\to 2}, \ldots, \mathsf{msg}_{2,n\to 2}). \end{aligned}$$

The adversary corrupting P_1 and P_2 can always compute both $y_{0,i}$ and $y_{1,i}$, since it can compute $\mathsf{msg}^{(b)}_{1,1\to 2}$, $\mathsf{msg}^{(b)}_{2,1\to 2}$ and $\mathsf{msg}^{(b)}_{1,2\to 2}$ for $b \in \{0, 1\}$ locally. Observe that, by correctness, $y_{0,2} = y^{(0)}$, and $y_{1,n} = y^{(1)}$.
For $b \in \{0, 1\}, j \in [3, \ldots, n]$ we also define values $y_{b,i,j}$ to be the output of P_2 computed without the second-round message from P_j.

$$\begin{aligned} y_{b,i,j} = \mathtt{output}_2(x_2, &\mathsf{msg}^{(b)}_{1,1\to 2}, \mathsf{msg}_{1,2\to 2}, \mathsf{msg}_{2,3\to 2}, \ldots, \mathsf{msg}_{1,n\to 2}, \\ &\mathsf{msg}^{(b)}_{2,1\to 2}, \mathsf{msg}^{(b)}_{2,2\to 2}, \mathsf{msg}'_{2,3\to 2}, \ldots, \mathsf{msg}'_{2,n\to 2}). \end{aligned}$$

where

$$\mathsf{msg}'_{2,j'\to 2} = \begin{cases} \mathsf{msg}_{2,j'\to 2}, & \text{if } j' \neq j, \\ \perp & \text{otherwise.} \end{cases}$$

By 2-strong last message resiliency, $y_{b,i} = y_{b,i,j}$ for all $j \in [3, \ldots, n]$. We also know that $y_{b,i-1,i} = y_{b,i,i}$, since in the computation of both of these values the second-round message from P_i is omitted, which is the only input that differs. By a combination of the previous two observations, we can conclude that $y_{b,i-1} = y_{b,i-1,i} = y_{b,i,i} = y_{b,i}$ for all $i \in [3, \dots, n]$. So, $y_{b,2} = \dots = y_{b,n}$.

By correctness, $y_{0,2} = y^{(0)}$. Since in strategy i the adversary corrupting P_1 and P_2 can always compute $y_{0,i} = y_{0,2}$, we know that in any strategy, the adversary can learn $y^{(0)}$. Similarly, by correctness, $y_{1,n} = y^{(1)}$; so, in any strategy, the adversary can learn $y^{(1)}$.

Corollary 4 (P2P-BC, FAIR, $t > 2$). *Secure computation of general functions with fairness cannot be achieved in two rounds the first of which is over peer-to-peer communication for corruption threshold $t > 2$.*

This corollary follows directly from Theorems 5 and 6.

4 Positive Results

In this section we describe our constructions. In Section 4.1 we list a few trivial observations, for completeness. In Section 4.3 we describe our constructions of secure computation with identifiable abort. In Section 4.2 we describe our construction of secure computation with guaranteed output delivery. Our constructions use some tools described in Appendix A.

4.1 Trivial Observations

Observation 1 *For $n = 2$, the possibility of selective abort implies the possibility of unanimous and identifiable abort.*

One could argue that Obs 1 should hold for $t = n - 1$, because there is only one honest party, so unanimity is trivial. However, we like to allow the adversary to corrupt fewer than t parties; if the adversary chooses to corrupt fewer than $t = n - 1$ parties, then there may be more than one honest party, making unanimity among them more challenging.

Observation 2 *The setting for t' inherits all positive results from the identical setting but with $t > t'$.*

4.2 Guaranteed Output Delivery

In this section, we describe a protocol that achieves guaranteed output delivery for $n > 2t$ in two rounds, with broadcast available in the second round only.

One could argue that, in the case of guaranteed output delivery, it does not matter whether the last round is over a broadcast channel or peer-to-peer channels.

Claim. Any protocol that achieves guaranteed output delivery against an adversary corrupting t parties in two broadcast rounds could be run with its second round over peer-to-peer channels and still achieve the same guarantee, but with $t - 1$ corruptions.

Proof (sketch). The adversary certainly can't abort honest parties by sending them incorrect things in the second round, since in a protocol with guaranteed output delivery, honest parties do not abort no matter what the adversary does. All the adversary could hope to do is cause disagreement, where different honest parties end up with output computed on different corrupt party inputs. However, if the adversary could do that, it could violate the security of the protocol by corrupting one additional honest party, computing different messages to it, and obtaining the output on two different sets of its own inputs.

We can conclude that any protocol in which such an attack is possible is insecure against $t+1$ corruptions; equivalently, any protocol which is secure against t corruptions does not permit such attacks in the second round when run with at most $t-1$ corruptions. So, any protocol that achieves guaranteed output delivery in two broadcast rounds against t corrupt parties also achieves guaranteed output delivery in one broadcast round followed by one round of peer-to-peer communication against $t-1$ corrupt parties.

Though this is a nice observation, the loss in corruption budget is dissatisfying. In this section, we prove that for any t, if there exists a protocol that achieves guaranteed output delivery against t corruptions in two broadcast rounds, there exists a protocol that achieves guaranteed output delivery against the *same number of corruptions* in two rounds only the first of which uses broadcast.

Our protocol $\Pi^{\mathsf{god}}_{\mathsf{bc-p2p}}$ follows the structure of the protocols described by Cohen *et al.* [CGZ20]. $\Pi^{\mathsf{god}}_{\mathsf{bc-p2p}}$ is a compiler that takes a protocol $\Pi^{\mathsf{god}}_{\mathsf{bc}}$ which achieves guaranteed output delivery given two rounds of broadcast (e.g., the protocol of [GLS15]), and achieves the same when broadcast is only available in the first round. Parties guarantee agreement on second-round messages by garbling and broadcasting their second-message functions, instead of sending the second messages peer-to-peer.

Notation: For every $i \in [n]$, denote by $\mathtt{C}_{i,x,r}(m_1,\dots,m_n)$ the Boolean circuit with hard-wired values x and r that upon receiving n inputs $m_1,\dots,m_n$, computes $\mathtt{snd\text{-}msg}_i$. For simplicity, assume that each first-round message is ℓ bits long, hence each such circuit has $L = n \cdot \ell$ inputs bits. We are assuming that the message $\bot$ is a fixed message of the appropriate length. Let g indicate the size of a garbling of $\mathtt{C}$.

Common input:

- A two-broadcast-round protocol $\Pi^{\mathsf{god}}_{\mathsf{bc}}$, represented by the set of functions $\{\mathtt{frst\text{-}msg}_i, \mathtt{snd\text{-}msg}_i, \mathtt{output}_i\}_{i\in[n]}$.
- A garbling scheme $(\mathtt{garble}, \mathtt{eval}, \mathtt{simGC})$.
- A CPA secure encryption scheme $(\mathtt{keygen}, \mathtt{enc}, \mathtt{dec})$.
- A t_{SS} out of n secret sharing scheme $(\mathtt{share}, \mathtt{reconstruct}, \mathtt{simshare})$ with $t_{SS} = \frac{n}{2}+1$.
- A non-interactive zero-knowledge proof system $(\mathtt{setup}, \mathtt{prove}, \mathtt{verify}, \mathtt{simP}, \mathtt{simP.Extract})$. We use this proof system for the following three relations.

$$\mathcal{R}_1 = \left\{ \begin{array}{l} \phi = \begin{pmatrix} m_j^1, c_{\mathtt{GC}}, \\ \{\tilde{k}_{i,l}^0, \tilde{k}_{i,l}^1\}_{i\in[n],l\in[L]}, \\ \{\mathtt{pk}_i\}_{i\in[n]} \end{pmatrix} \\ w = \begin{pmatrix} \mathtt{C}_{j,x,r}, \mathtt{GC}, \\ o_{\mathtt{GC}}, (x,r), R_{\mathtt{GC}}, R_{\mathtt{ss}}, \\ \{R_{i,l}^0, R_{i,l}^1\}_{i\in[n],l\in[L]} \end{pmatrix} \end{array} \middle| \begin{array}{l} m_j^1 = \mathtt{frst\text{-}msg}_j(x,r) \\ \wedge(\mathtt{GC}, \{K_l^0, K_l^1\}_{l\in[L]}) = \mathtt{garble}(1^\lambda, \mathtt{C}; R_{\mathtt{GC}}) \\ \wedge c_{\mathtt{GC}}, o_{\mathtt{GC}} = \mathtt{commit}(\mathtt{GC}) \\ \wedge\{\{\{k_{j,l}^b\}_{j\in[n]} = \mathtt{share}(K_l^b; R_{\mathtt{SS}})\}_{b\in\{0,1\}}\}_{l\in[L]} \\ \wedge\{\{\{\tilde{k}_{i,l}^b = \mathtt{enc}(\mathtt{pk}_i, k_{i,l}^b; R_{i,l}^b)\}_{b\in\{0,1\}}\}_{l\in[L]}\}_{i\in[n]} \end{array} \right\},$$

$$\mathcal{R}_2 = \{\phi = (\mathtt{pk}, c, m), w = (\mathtt{sk}, o_{\mathtt{GC}}) | m = \mathtt{dec}(\mathtt{sk}, c)\}.$$

$$\mathcal{R}_3 = \{\phi = (c_{\mathtt{GC}}, \mathtt{GC}), w = o_{\mathtt{GC}} | \mathtt{GC}, o_{\mathtt{GC}} = \mathtt{open}(c_{\mathtt{GC}})\}.$$

Protocol $\Pi^{\mathsf{god}}_{\mathsf{bc-p2p}}$ with $n = 2t+1$

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

Setup. 1. Set up the PKI and publish for party P_i her public key $\mathtt{pk}_i$.

2. Set up the common reference strings $crs_1 \leftarrow \mathtt{setup}(1^\lambda, \mathcal{R}_1)$, $crs_2 \leftarrow \mathtt{setup}(1^\lambda, \mathcal{R}_2)$ and $crs_3 \leftarrow \mathtt{setup}(1^\lambda, \mathcal{R}_3)$ for the zero knowledge proof system.
3. Set up the correlated randomness $(r_1, \ldots, r_n) \leftarrow \mathcal{D}^{\mathsf{bc}}_{\mathsf{corr}}$ and distribute r_i to party P_i for $i \in [n]$.

First round. Every party P_i proceeds as follows:

1. Let $\mathsf{msg}_i^1 = \mathtt{frst\text{-}msg}_i(x_i, r_i)$ be P_i's first-broadcast-round message in $\Pi^{\mathsf{god}}_{\mathsf{bc}}$.
2. Using fresh randomness $R_{\mathtt{GC}_i}$, compute $(\mathtt{GC}_i, \vec{K}_i) = \mathtt{garble}(1^\lambda, \mathtt{C}_{i,x_i,r_i}; R_{\mathtt{GC}_i})$, where $\vec{K}_i = (K_1^0, K_1^1, \ldots, K_L^0, K_L^1)$.
3. For every $l \in L$ and $b \in \{0,1\}$, using fresh randomness $R_{\mathtt{ss}_i}$, compute $(k_{i\to 1,l}^b, \ldots, k_{i\to n,l}^b) = \mathtt{share}(1^\lambda, K_l^b; R_{\mathtt{ss}_i})$.
4. $c_{\mathtt{GC}_i}, o_{\mathtt{GC}_i} \leftarrow \mathtt{commit}(\mathtt{GC}_i)$.
5. For every $l \in L$, $j \in [n]$ and $b \in \{0,1\}$, using fresh randomness $R_{i\to j,l}^b$, compute $\tilde{k}_{i\to j,l}^b = \mathtt{enc}(\mathtt{pk}_j, k_{i\to j,l}^b; R_{i\to j,l}^b)$.
6. Compute a zero knowledge proof of correct behavior:
   (a) Set $\phi_i = \Big(\mathsf{msg}_i^1, c_{\mathtt{GC}_i}, \{\tilde{k}_{i\to j,l}^0, \tilde{k}_{i\to j,l}^1\}_{j\in[n],l\in[L]}, \{\mathtt{pk}_j\}_{j\in[n]}\Big)$ and
   $w_i = \Big(\mathtt{GC}_i, o_{\mathtt{GC}_i}, \mathtt{C}_{i,x_i,r_i}, (x_i, r_i), R_{\mathtt{GC}_i}, R_{\mathtt{ss}_i}, \{R_{i,l}^0, R_{i,l}^1\}_{i\in[n],l\in[L]}\Big)$.
   (b) Run $\pi_i \leftarrow \mathtt{prove}(crs_1, \phi_i, w_i)$.
7. Broadcast $(\mathsf{msg}_i^1, c_{\mathtt{GC}_i}, \{\tilde{k}_{i\to j,l}^0, \tilde{k}_{i\to j,l}^1\}_{l\in[L],j\in[n]}, \pi_i)$.

Second round. Every party P_i proceeds as follows:

1. Let $(\mathsf{msg}_j^1, c_{\mathtt{GC}_j}, \{\tilde{k}_{j\to i,l}^0, \tilde{k}_{j\to i,l}^1\}_{l\in[L]}, \pi_j)$ be the first round received from party P_j.
2. Let $\mathtt{NA}$ be the set of indices for which $\mathtt{verify}(crs_1, \phi_j, \pi_j) = 1$.
3. For every $j \notin \mathtt{NA}$ set $\mathsf{msg}_j^1 = \bot$.
4. Denote the concatenation of all the messages msg_j^1 for $j \in [n]$ as

$$(\nu_1, \ldots, \nu_L) = (\mathsf{msg}_1^1, \ldots, \mathsf{msg}_n^1) \in \{0,1\}^L$$

5. For $l \in [L]$ and $j \in \texttt{NA}$ run $k_{j\to i,l} \leftarrow \texttt{dec}(\texttt{sk}_i, \tilde{k}^{\nu_l}_{j_i\to i,l})$, and $\pi_{j\to i,l} \leftarrow \texttt{prove}\Big(crs_2, \phi = (\texttt{pk}_i, \tilde{k}^{\nu_l}_{j\to i,l}, k_{j\to i,l}), w = \texttt{sk}_i\Big)$.
6. Set $\phi_{\texttt{GC}_i} = (c_{\texttt{GC}_i}, \texttt{GC}_i), w_{\texttt{GC}_i} = (o_{\texttt{GC}_i})$ run $\pi_{\texttt{GC}_i} \leftarrow \texttt{prove}(crs_3, \phi_{\texttt{GC}_i}, w_{\texttt{GC}_i})$.
7. Send to all the parties the message $\{\texttt{GC}_i, \pi_{\texttt{GC}_i}, \{k_{j\to i,l}, \pi_{j\to i,l}\}_{l\in[L]}\}_{j\in\texttt{NA}}$.

Output. Every party P_i proceeds as follows:
1. Let $\{\{k_{j'\to j,l}, \pi_{j'\to j,l}\}_{l\in[L]}\}_{j'\in\texttt{NA}}$ be the second-round message received from party P_j.
2. Let $\texttt{NA}'$ be the set of indices j for which $\texttt{verify}\Big(crs_2, \phi = (c_{\texttt{GC}_i}, \texttt{GC}_i, \texttt{pk}_j, \tilde{k}_{j'\to j,l}, k_{j'\to j,l}), \pi_{j'\to j,l}\Big) = 1$ and $\texttt{verify}(crs_3, \phi_{\texttt{GC}_{j'}} = (\texttt{GC}_{j'}, c_{\texttt{GC}_{j'}})) = 1$ for all $j' \in \texttt{NA}$.
3. For every $j \in \texttt{NA}'$, for every $l \in [L]$ compute $K_{j,l} = \texttt{reconstruct}(\{k_{j\to j',l}\}_{j'\in\texttt{NA}'})$.
4. For every $j \in \texttt{NA}'$, evaluate the garbled circuit received from P_j as $\mathsf{msg}^2_j = \texttt{eval}(\texttt{GC}_j, K_{j,1}, \ldots, K_{j,L})$.
5. For every $j \notin \texttt{NA}'$ set $\mathsf{msg}^2_j = \bot$.
6. Output $y = \texttt{output}_i(x_i, r_i, (\mathsf{msg}^1_1, \ldots, \mathsf{msg}^1_n), (\mathsf{msg}^2_1, \ldots, \mathsf{msg}^2_n))$.

Theorem 7 (BC-P2P, GOD, $n > 2t$). *Let $\mathcal{F}$ be an efficiently computable n-party function and let $n > 2t$. Let $\Pi^{\mathsf{god}}_{\mathsf{bc}}$ be a two broadcast-round protocol that securely computes $\mathcal{F}$ with guaranteed output delivery with a black-box straight-line simulator. Assume that* $(\texttt{garble}, \texttt{eval}, \texttt{simGC})$ *is a secure garbling scheme,* $(\texttt{share}, \texttt{reconstruct}, \texttt{simshare})$ *is a secure secret sharing scheme with threshold $\frac{n}{2}+1$,* $(\texttt{keygen}, \texttt{enc}, \texttt{dec})$ *is a CPA secure encryption scheme, and* $(\texttt{setup}, \texttt{prove}, \texttt{verify}, \texttt{simP}, \texttt{simP.Extract})$ *is a non-interactive simulation-extractable zero-knowledge proof system. Then, $\Pi^{\mathsf{god}}_{\mathsf{bc-p2p}}$ securely computes $\mathcal{F}$ with guaranteed output delivery in two rounds, the first of which is over a broadcast channel, and the second of which is over peer-to-peer channels.*

The simulator. Let $\mathcal{S}^{\mathsf{god}}_{\mathsf{bc}}$ be the simulator for $\Pi^{\mathsf{god}}_{\mathsf{bc}}$. The simulator $\mathcal{S}$ proceeds as follows:

Simulator $\mathcal{S}^{\mathsf{god}}_{\mathsf{bc-p2p}}$

Setup. 1. $\mathcal{S}$ sets up a PKI and runs $(crs_1, td_1) \leftarrow \texttt{setup}(1^\lambda)$, $(crs_2, td_2) \leftarrow \texttt{setup}(1^\lambda)$ and $(crs_3, td_3) \leftarrow \texttt{setup}(1^\lambda)$.
2. $\mathcal{S}$ invokes $\mathcal{A}$ on her inputs and the simulated correlated randomness for corrupted parties.

First round. $\mathcal{S}$, on behalf of the honest party P_h for all $h \in \mathbb{H}$, does the following steps:
1. For every $l \in L$ and $b \in \{0,1\}$, $\mathcal{S}$ computes $k^b_{h\to 1,l}, \ldots, k^b_{h\to n,l} \leftarrow \texttt{share}(0^L)$.

2. For every $l \in L$, $j \in [n]$, $\mathcal{S}$ computes $\tilde{k}^b_{h\to j,l} \leftarrow \mathtt{enc}(\mathtt{pk}_j, k^b_{h\to j,l})$.
3. $c_{\mathtt{GC}_h}, o_{\mathtt{GC}_h} \leftarrow \mathtt{commit}(0^g)$.
4. $\mathcal{S}$ invokes $\mathcal{S}^{\mathsf{god}}_{\mathsf{bc}}$ in order to get first round message $\hat{\mathsf{msg}}^1_h$ of $\Pi_{\mathtt{bc}}$.
5. Let $\phi_h = \Big(\hat{\mathsf{msg}}^1_h, c_{\mathtt{GC}_h}, \{\tilde{k}^0_{h\to j,l}, \tilde{k}^1_{h\to j,l}\}_{j\in[n],l\in[L]}, \{\mathtt{pk}_j\}_{j\in[n]}\Big)$. $\mathcal{S}$ runs $\pi_h \leftarrow \mathtt{simP}(crs_1, td_1, \phi_h)$.
6. $\mathcal{S}$ broadcasts $(\hat{\mathsf{msg}}^1_h, c_{\mathtt{GC}_h}, \{\tilde{k}^0_{h\to j,l}, \tilde{k}^1_{h\to j,l}\}_{l\in[L],j\in[n]}, \pi_h)$.

Second round. Let $\{(\hat{\mathsf{msg}}^1_a, \mathtt{GC}_j, \{\tilde{k}^0_{a\to j,l}, \tilde{k}^1_{a\to j,l}\}_{l\in[L],j\in[n]}, \pi_a)\}_{a\in\mathbb{A}}$ be the set of messages received from $\mathcal{A}$. $\mathcal{S}$, on behalf of the honest party P_h for all $h \in \mathbb{H}$, does the following steps:

1. For $a \in \mathbb{A}$, $\mathcal{S}$ runs $\mathtt{simP.Extract}(crs_1, \phi_a, \pi_a)$. If one of the extraction fails, $\mathcal{S}$ outputs $\mathtt{abort}$.
2. Let $\mathtt{NA}$ be the set of indices for which the output of $\mathtt{simP.Extract}$ matches the inputs of dishonest parties obtained from $\mathcal{S}^{\mathsf{god}}_{\mathsf{bc}}$. Let $\mathtt{NA} = \mathtt{NA} \cup \mathbb{H}$. For every $a \notin \mathtt{NA}$, $\mathcal{S}$ sets $\mathsf{msg}^1_a = \perp$.
3. $\mathcal{S}$ feeds messages $\{\hat{\mathsf{msg}}^1_a\}_{a\in\mathbb{A}}$ to $\mathcal{S}^{\mathsf{god}}_{\mathsf{bc}}$ receiving in response the corrupt parties' inputs $\vec{x} = \{x_i\}_{i\in\mathbb{A}}$. $\mathcal{S}$ simulates the interaction between $\mathcal{S}^{\mathsf{god}}_{\mathsf{bc}}$ and the trusted third party that computes $\mathcal{F}$. Specifically, $\mathcal{S}$ forwards the message that she received from $\mathcal{S}^{\mathsf{god}}_{\mathsf{bc}}$ to the trusted third party, and she receives back y. $\mathcal{S}$ forwards y to $\mathcal{S}^{\mathsf{god}}_{\mathsf{bc}}$, which outputs honest messages $\{\hat{\mathsf{msg}}^2_h\}_{h\in\mathbb{H}}$.
4. Let
$$(\nu_1, \ldots, \nu_L) = (\hat{\mathsf{msg}}^1_1, \ldots, \hat{\mathsf{msg}}^1_n) \in \{0,1\}^L.$$
5. For every $l \in [L]$, $\mathcal{S}$ runs $\{k^{\nu_l}_{h\to h',l}\}_{h'\in\mathbb{H}} \leftarrow \mathtt{simshare}(K_{h,l}, \{k^{\nu_l}_{h\to a,l}\}_{a\in\mathbb{A}})$.
6. Let $\mathtt{C}_h$ be the circuit computing $\mathtt{snd\text{-}msg}_h$ with input and randomness set to 0. $\mathcal{S}$ runs $(\mathtt{GC}_h, K_{h,1}, \ldots, K_{h,L}) \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}_h, \hat{\mathsf{msg}}^2_h)$. Run $\pi_{\mathtt{GC}_h} \leftarrow \mathtt{simP}(crs_3, td_3, (c_{\mathtt{GC}_h}, \mathtt{GC}_h))$.
7. For $l \in [L]$ and $h' \in \mathtt{NA} \cap \mathbb{H}$, $\mathcal{S}$ runs $\pi^{\nu_l}_{h'\to h,l} \leftarrow \mathtt{simP}\Big(crs_2, td_2, (\mathtt{pk}_h, \tilde{k}^{\nu_l}_{h'\to h,l}, k^{\nu_l}_{h'\to h,l})\Big)$.
8. For $l \in [L]$ and $a \in \mathtt{NA} \cap \mathbb{A}$, $\mathcal{S}$ honestly runs $k^{\nu_l}_{a\to h} \leftarrow \mathtt{dec}(\mathtt{sk}_h, \tilde{k}^{\nu_l}_{a\to h,l})$, and $\pi^{\nu_l}_{a\to h,l} \leftarrow \mathtt{prove}\Big(crs_2, td_2, \phi = (\mathtt{pk}_h, \tilde{k}^{\nu_l}_{a\to h,l}, k^{\nu_l}_{a\to h,l}), w = \mathtt{sk}_h\Big)$.
9. $\mathcal{S}$ sends to $\mathcal{A}$ the message $(\pi_{\mathtt{GC}_h}, \{k^{\nu_l}_{j\to h,l}, \pi^{\nu_l}_{j\to h,l}\}_{l\in[L],j\in[n]})$.

Output. $\mathcal{S}$ outputs the output of $\mathcal{A}$ and terminates.

We will now proceeds through a series of hybrid experiments in oder to prove that the joint distribution of the output of $\mathcal{A}$ and the outputs of the honest parties in the ideal execution is computationally indistinguishable from the joint distribution of the output of $\mathcal{A}$ and the outputs of the honest parties in a real protocol execution. The hybrid experiments are listed below. The output of each experiment is defined as the output of $\mathcal{A}$ and the output of the honest parties.

- $\mathtt{Expt}^0_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$: In this experiment, the simulator $\mathcal{S}_0$ has access to the internal state of the trusted party computing $\mathcal{F}$. Therefore $\mathcal{S}_0$ chooses the output values of the honest parties. In the execution of $\Pi_{\mathtt{bc-p2p}}$ the simulator is interacting with $\mathcal{A}$ on behalf of the honest parties. The output of this hybrid experiment is the output of the honest parties and the output of $\mathcal{A}$ in the execution of $\Pi_{\mathtt{bc-p2p}}$ explained above. It follow trivially that the output of $\mathtt{Expt}^0_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ and the the output of the real world experiment are identically distributed.
- $\mathtt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$: in this experiment $\mathtt{Expt}^0_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ is modified as follows. In order to compute π_h in the first round $\mathcal{S}_1$ runs $\mathtt{simP}(crs_1, td_1, \phi_h)$, for $h \in \mathbb{H}$. Moreover $\mathcal{S}_1$ executes the step 1 of $\mathcal{S}$.

  Claim. $\mathtt{Expt}^0_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ and $\mathtt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ are computationally indistinguishable.

  Proof (Sketch). The proof proceeds via $|\mathbb{H}|+1$ hybrids arguments: in the j-th hybrid experiment π_h of honest party P_h with $h \leq j$ are simulated as in $\mathtt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ and for $h > j$ are computed as $\mathtt{Expt}^0_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$. In order to claim that two neighboring hybrids are computationally indistinguishable we can rely on the simulation extractability of the zero-knowledge proof system. Notice that the abort probability between two neighboring hybrids increases only of a negligible amount due to simulation extractability of the zero-knowledge proof system. The proof conclude observing that the 0-th hybrid corresponds to $\mathtt{Expt}^0_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ and the $|\mathbb{H}|$-th corresponds to $\mathtt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$.
- $\mathtt{Expt}^2_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$: in this experiment $\mathtt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ is modified as follows. In the second round, for all $h \in \mathbb{H}$, $\mathcal{S}_2$ runs $\pi_{\mathtt{GC}_h} \leftarrow \mathtt{simP}(crs_3, td_3, (c_{\mathtt{GC}_h}, \mathtt{GC}_h))$ where $\mathtt{GC}_h$ is an honestly generated garbled circuit like in $\Pi_{\mathtt{bc-p2p}}$. Moreover $\mathcal{S}_3$ for all $j \in \mathtt{SA}$ runs $\mathtt{simP.Extract}(crs_3, (\pi_{\mathtt{GC}_j}))$ if the extraction fails output $\mathtt{abort}$, where $\pi_{\mathtt{GC}_j}$ is sent by corrupted party j and $\mathtt{SA}$ is the set of corrupted party that send a second round message.

  Claim. $\mathtt{Expt}^2_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ and $\mathtt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ are computationally indistinguishable.

  Proof (Sketch). The proof proceeds in a similar way to the previous one.
- $\mathtt{Expt}^3_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$: in this experiment $\mathtt{Expt}^2_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ is modified as follows. Let $(\nu_{i,1}, \ldots, \nu_{i,L})$ be computed as in step 4 of $\Pi_{\mathtt{bc-p2p}}$ (where $\mathtt{NA}$ is computed as in $\mathtt{Expt}^2_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$). In the second round $\mathcal{S}_3$ runs $\pi^{\nu_l}_{h'\to h,l} \leftarrow \mathtt{simP}\Big(crs_2, td_2, (\mathtt{pk}_h, \tilde{k}^{\nu_l}_{h'\to h,l}, k^{\nu_l}_{h'\to h,l})\Big)$, for $h, h' \in \mathbb{H}$. Moreover for $h \in \mathbb{H}$ $j \in \mathtt{SA}$ for $l \in [L]$ $\mathcal{S}_3$ runs $\mathtt{simP.Extract}(crs_2, (\tilde{k}^{\nu_{j,l}}_{h\to j,l}, k_{h\to j,l}), \pi_{h\to j,l})$ where $\tilde{k}^{\nu_{j,l}}_{h\to j,l}, k_{h\to j,l}$ is sent by corrupted party j and $\mathtt{SA}$ is the set of corrupted parties that send a second round message.

  Claim. $\mathtt{Expt}^3_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ and $\mathtt{Expt}^2_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ are computationally indistinguishable.

Proof (Sketch). The proof proceeds in a similar way to the previous one.

– $\mathtt{Expt}^4_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$: in this experiment $\mathtt{Expt}^3_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ is modified as follows. In the first round, the simulator $\mathcal{S}_4$ sends a commitment to 0^g for each honest party. In more detail, $\mathcal{S}_4$ computes $c_{\mathtt{GC}_h}, o_{\mathtt{GC}_h} \leftarrow \mathtt{commit}(0^g)$ for all $h \in \mathbb{H}$.

Claim. $\mathtt{Expt}^4_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ and $\mathtt{Expt}^3_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ are computationally indistinguishable.

Proof (Sketch). The proof proceeds via $|\mathbb{H}| + 1$ hybrids arguments. In the j-th hybrid experiment the commitments $c_{\mathtt{GC}_h}$ of the first j honest parties P_h are commitments to 0^g as in $\mathtt{Expt}^4_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$; for the other $|\mathbb{H}| - j$ honest parties, the commitments are computed as in $\mathtt{Expt}^3_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$. In order to claim that two neighboring hybrids are computationally indistinguishable we can rely on the hiding property of the commitment scheme. The proof concludes by observing that the 0-th hybrid corresponds to $\mathtt{Expt}^3_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ and the $|\mathbb{H}|$-th corresponds to $\mathtt{Expt}^4_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$.

– $\mathtt{Expt}^5_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$: in this experiment $\mathtt{Expt}^4_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ is modified as follows. The simulator $\mathcal{S}_5$ sends in the first round encryptions of 0^L to all the honest parties. In more detail, for every $l \in L$, $h, h' \in \mathbb{H}$ and $b \in \{0, 1\}$, $\mathcal{S}_5$ computes $\tilde{k}^b_{h \to h',l} \leftarrow \mathtt{enc}(\mathtt{pk}_{h'}, 0^L)$.

Claim. $\mathtt{Expt}^5_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ and $\mathtt{Expt}^4_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ are computationally indistinguishable.

Proof (Sketch). The proof proceeds via $2|\mathbb{H}|^2 l + 1$ hybrids arguments. For each $h, h' \in \mathbb{H}$, $l \in [L]$, $b \in \{0, 1\}$, we define hybrid experiment $\mathtt{Expt}_{h \to h',l,b}$. In each subsequent experiment $\mathtt{Expt}_{h \to h',l,b}$, the encryption $\tilde{k}^b_{h \to h',l'}$ becomes an encryption of 0^L. In order to claim that two neighboring hybrids are computationally indistinguishable we can rely on the CPA security of the encryption scheme. The proof concludes observing that the 0-th hybrid corresponds to $\mathtt{Expt}^4_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ and the last hybrid corresponds to $\mathtt{Expt}^5_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$.

– $\mathtt{Expt}^6_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$: in this experiment $\mathtt{Expt}^5_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ is modified as follows. The shares are computed differently; in more detail, the simulator $\mathcal{S}_6$ computes the shares on behalf of the honest parties according to step 3a of the ideal world simulator $\mathcal{S}$ described above.
Let $(\nu_{i,1}, \ldots, \nu_{i,L})$ be computed as in step 4 of $\Pi_{\mathtt{bc-p2p}}$ (where $\mathtt{NA}$ is computed as before). $\mathcal{S}_6$ computes for every $l \in [L]$ for all $h \in \mathbb{H}$ $k^{\nu_{h,l}}_{h \to i_1,l}, \ldots, k^{\nu_{h,l}}_{h \to i_{|\mathbb{H}|},l} \leftarrow \mathtt{simshare}(K_{h,l}, k^{\nu_{h,l}}_{h \to i_1,l}, \ldots, k^{\nu_{h,l}}_{h \to i_{|\mathbb{A}|},l})$ (where $K_{h,l}$ are computed as in step 3 of $\Pi_{\mathtt{bc-p2p}}$).

Claim. $\mathtt{Expt}^6_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ and $\mathtt{Expt}^5_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ are perfect indistinguishable.

Proof (Sketch). Note that in the first round both in $\mathtt{Expt}^6_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ that in $\mathtt{Expt}^5_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ the honest parties send encryptions of 0^L to all the other honest parties. To reconstruct a shared label $\mathcal{A}$ needs t_{SS} shares, since by assumption at least t_{SS} of the parties are honest therefore $\mathcal{A}$ at the end of the first round can not reconstruct any secret. Due to the security of the secret sharing scheme $\mathtt{Expt}^6_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ and $\mathtt{Expt}^5_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ are perfect indistinguishable.

– $\mathtt{Expt}^7_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$: this experiment proceeds as the experiment $\mathtt{Expt}^6_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ except that the garble circuits regarding the honest parties are computed using the simulated procedure $\mathtt{simGC}$. Let $\mathtt{C}_{h,x_h,r_h}$ be the circuit $\mathtt{snd\text{-}msg}_h$ with hard-wired input x_h and randomness set to r_h. $\mathcal{S}_7$ executes, for all $h \in \mathbb{H}$, $(\mathtt{GC}_h, K_{h,1}, \ldots, K_{h,L}) \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}_{h,x_h,r_h}, \mathsf{msg}^2_h)$, where msg^2_h is the message computed by P_h in the execution of $\Pi_{\mathtt{bc-p2p}}$.

Claim. $\mathtt{Expt}^6_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ and $\mathtt{Expt}^7_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ are computationally indistinguishable.

Proof (Sketch). The proof proceeds via $|\mathbb{H}| + 1$ hybrids arguments: in the j-th hybrid experiment the garble circuit of honest party P_h with $h \leq j$ are simulated as in $\mathtt{Expt}^7_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ and for $h > j$ are computed as $\mathtt{Expt}^6_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$. In order to claim that two neighboring hybrids are computationally indistinguishable we can rely on security of garbling scheme. The proof conclude observing that the 0-th hybrid corresponds to $\mathtt{Expt}^6_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ and the $|\mathbb{H}|$-th corresponds to $\mathtt{Expt}^7_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$.

– $\mathtt{Expt}^8_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$: this experiment proceeds as the experiment $\mathtt{Expt}^7_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ except that the garble circuits regarding the honest parties are computed using the simulated procedure $\mathtt{simGC}$. Let $\mathtt{C}_h$ be the circuit $\mathtt{snd\text{-}msg}_h$ with hard-wired input and randomness set to 0. $\mathcal{S}_8$ executes, for all $h \in \mathbb{H}$, $(\mathtt{GC}_h, K_{h,1}, \ldots, K_{h,L}) \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}_h, \mathsf{msg}^2_h)$, where msg^2_h is the message computed by P_h in the execution of $\Pi_{\mathtt{bc-p2p}}$.

Claim. $\mathtt{Expt}^8_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ and $\mathtt{Expt}^7_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ are computationally indistinguishable.

Proof (Sketch). The proof proceeds similar to the proof of previous claim.

– $\mathtt{Expt}^9_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$: this experiment proceeds as the experiment $\mathtt{Expt}^8_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ except that instead of computing the messages of $\Pi^{\mathsf{god}}_{\mathsf{bc}}$ relaying on honest parties inputs, the simulator $\mathcal{S}_9$ uses the messages given in output by $\mathcal{S}^{\mathsf{god}}_{\mathsf{bc}}$.

Claim. $\mathtt{Expt}^8_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ and $\mathtt{Expt}^9_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ are computationally indistinguishable.

The indistinguishability between $\mathtt{Expt}^8_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ and $\mathtt{Expt}^9_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$ follows from the security of $\Pi^{\mathsf{god}}_{\mathsf{bc}}$. More in details, any distinguisher between $\mathtt{Expt}^8_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{bc-p2p}}}$

and $\mathtt{Expt}^{9}_{\mathbb{A},\mathcal{A},\Pi_{\mathsf{bc-p2p}}}$ can be used to distinguish between $\mathcal{S}^{\mathsf{god}}_{\mathsf{bc}}$ and real world execution of $\Pi^{\mathsf{god}}_{\mathsf{bc}}$. Observe that the reduction goes thought because already in $\mathtt{Expt}^{8}_{\mathbb{A},\mathcal{A},\Pi_{\mathsf{bc-p2p}}}$ the probability that the experiment aborts is negligible. The proof crucially rely on the ability if the simulator $\mathcal{S}^{\mathsf{god}}_{\mathsf{bc}}$ to extract the adversary's input from her first round of Π_{bc} (which is sent over broadcast channel in $\Pi_{\mathsf{bc-p2p}}$), and this is implied from the assumption that $\mathcal{S}^{\mathsf{god}}_{\mathsf{bc}}$ is strait-line and black-box.

The proof ends observing that in $\mathtt{Expt}^{9}_{\mathbb{A},\mathcal{A},\Pi_{\mathsf{bc-p2p}}}$ $\mathcal{S}_9$ does not need anymore to have access to the internal state of the trusted third part that computes $\mathcal{F}$ and therefore $\mathtt{Expt}^{9}_{\mathbb{A},\mathcal{A},\Pi_{\mathsf{bc-p2p}}}$ and the ideal world experiment are identically distributed.

4.3 Identifiable Abort

In this section we show two protocols achieving secure computation with identifiable abort in two rounds, with the first round only using peer-to-peer channels. Our first protocol requires that $t < \frac{n}{3}$; the second requires that $t < \frac{n}{2}$.

Our first protocol follows the structure of the protocols described by Cohen *et al.* [CGZ20]. It is described as a compiler that takes a protocol Π_{bc} which achieves the desired guarantees given two rounds of broadcast, and achieves those same guarantees in the broadcast pattern we are interested in, which has broadcast available in the second round only. However, using the techniques of Cohen *et al.* when limited to peer-to-peer channels in the first round is tricky. One of the major challenges is that, in the techniques of Cohen *et al.*, in the second round, every party P_i must forward to everyone else exactly one of a pair of objects (shares of labels) which P_i should have obtained from every other party P_j. However, since the first round is over peer-to-peer channels, P_i can claim that it didn't get the shares of labels from P_j, and the computation must still complete, since it is unclear who to blame — P_i or P_j. We get around this by requiring P_i to publish a shared key with P_j if P_j didn't send anything in the first round. P_j must always broadcast encryptions of its label shares in the second round; the shared key guarantees that either party P_i is able to forward the appropriate share in the second round, or everyone is able to obtain both shares by using the published key. If P_i and P_j were both honest, the adversary obtaining both shares could lead to a violation of privacy; however, if the key is published, we know that either P_i or P_j were corrupt, so the adversary knows both shares anyway.

We next show that these techniques cannot be applied when $n \leq 3t$; we sketch a second protocol that uses stronger assumptions such as obfuscation, or correlated randomness, to achieve identifiable abort for $\frac{n}{3} \leq t < \frac{n}{2}$.

P2P-BC Identifiable Abort with $n > 3t$ Our first protocol $\Pi^{\mathsf{id\text{-}abort}}_{\mathsf{p2p-bc}}$ achieves security with identifiable abort when $n > 3t$ and the first round is over peer-to-peer channels. $\Pi^{\mathsf{id\text{-}abort}}_{\mathsf{p2p-bc}}$, which follows the structure of the protocols described by Cohen *et al.* [CGZ20], is formally described in Figure 4.3 borrowing some notation from Cohen *et al.* It uses the following:

Common Input:

- A two-broadcast-round protocol $\Pi_{\texttt{bc}}$, represented by the set of functions $\{\texttt{frst-msg}_i, \texttt{snd-msg}_i, \texttt{output}_i\}_{i\in[n]}$.
- A garbling scheme $(\texttt{garble}, \texttt{eval}, \texttt{simGC})$.
- A t_{SS}-out-of-n secret sharing scheme $(\texttt{share}, \texttt{reconstruct}, \texttt{simshare})$ with $t_{SS} = \frac{2n}{3} + 1$.
- A symmetric encryption scheme $(\texttt{enc}, \texttt{dec})$.
- A commitment scheme $(\texttt{commit}, \texttt{open})$.
- A non-interactive key agreement scheme $(\texttt{keygen}, \texttt{keyagree})$.
- A non-interactive zero-knowledge proof system $(\texttt{setup}, \texttt{prove}, \texttt{verify}, \texttt{simP}, \texttt{simP.Extract})$ for the following relations:

$$\mathcal{R}_1 = \left\{ \begin{array}{l|l} & \texttt{sk}_i \text{ is the secret key corresp. to } \texttt{pk}_i \\ & \wedge(\texttt{GC}, \{K_l^0, K_l^1\}_{l\in[L]}) = \texttt{garble}(1^\lambda, \texttt{C}; R) \\ & \wedge\texttt{open}(c, o) = R \\ \phi = \begin{pmatrix} \{\texttt{pk}_j\}_{j\in[n]}, c, c_{\texttt{GC}}, \\ \{\overline{k}_{j,l}^0, \overline{k}_{j,l}^1\}_{l\in[k+1,\dots,L]}\}_{j\in[n]} \end{pmatrix} & \wedge\texttt{open}(c_{\texttt{GC}}, o_{\texttt{GC}}) = \texttt{GC} \\ w = (\texttt{C}, \texttt{GC}, R, o_{\texttt{GC}}, o, \texttt{sk}_i) & \wedge\{\{\{k_{j,l}^b\}_{j\in[n]} = \\ & \ \texttt{share}(K_l^b; R)\}_{b\in\{0,1\}}\}_{l\in[k+1,\dots,L]} \\ & \wedge\texttt{k}_j = \texttt{keyagree}(\texttt{pk}_j, \texttt{sk}_i) \\ & \wedge\{\{\{\overline{k}_{j,1}^b = \\ & \ \texttt{enc}(\texttt{k}_j, k_{j,l}^b; R)\}_{b\in\{0,1\}}\}_{l\in[k+1,\dots,L]}\}_{j\in[n]} \end{array} \right\},$$

$$\mathcal{R}_2 = \left\{ \begin{array}{l|l} \phi = \begin{pmatrix} \texttt{pk}_i, \texttt{pk}_j, \\ \{\overline{k}^l, k^l\}_{l\in k+1,\dots,L} \end{pmatrix} & \texttt{sk}_i \text{ is the secret key corresp. to } \texttt{pk}_i \\ & \wedge\ \texttt{k} = \texttt{keyagree}(\texttt{pk}_j, \texttt{sk}_i) \\ w = \texttt{sk}_i & \wedge\ \{\overline{k}^l = \texttt{enc}(\texttt{k}, k^l)\}_{l\in[k+1,\dots,L]} \end{array} \right\},$$

$$\mathcal{R}_3 = \left\{ \begin{array}{l|l} \phi = (\texttt{pk}_i, \texttt{pk}_j, \texttt{k}) & \texttt{sk}_i \text{ is the secret key corresp. to } \texttt{pk}_i \\ w = (\texttt{sk}_i) & \wedge\texttt{k} = \texttt{keyagree}(\texttt{pk}_j, \texttt{sk}_i) \end{array} \right\}.$$

Private Input: Each party P_i has a private input $x_i \in \{0,1\}^*$.
Public Key Infrastructure: Each party P_i has a secret key $\texttt{sk}_i$ the associated public key $\texttt{pk}_i$ for which is known; these are used for key agreement. Each party also holds randomness R_i to which a commitment $c_{r,i}$ is known. (P_i also knows the associated opening o_i.) For notational simplicity, we let P_i re-use this same randomness R_i in several places; we implicitly assume that P_i expands R_i to obtain distinct randomness by using a PRF in the appropriate way.
Correlated randomness: The correlated randomness $(r_1, \dots, r_n) \leftarrow \mathcal{D}_{\texttt{corr}}^{\texttt{bc}}$ (for $\Pi_{\texttt{bc}}$) is sampled at the onset of the protocol, and every party P_i receives r_i.
Notation: For every $i \in [n]$, denote by $\texttt{C}_i(x_i, r_i, m_1, \dots, m_n)$ the boolean circuit that takes as input P_i's input x_i, randomness r_i, and first-round messages $m_1, \dots, m_n$, and computes $\texttt{snd-msg}_i$. For simplicity, assume that (x_i, r_i) together are k bits long, and each first-round message is l bits long, so each such circuit has $L = k + n \cdot l$ inputs bits.
Note that $\texttt{C}_i$ is public; let g be the size of a garbled $\texttt{C}_i$.

Protocol $\Pi^{\text{id-abort}}_{\text{p2p-bc}}$ with $n = 3t + 1$

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

1. Set up the PKI and publish for party P_i her public key $(\mathtt{pk}_i, c_{r,i})$.
2. Set up the common reference strings $crs_1 \leftarrow \mathtt{setup}(1^\lambda, \mathcal{R}_1)$, $crs_2 \leftarrow \mathtt{setup}(1^\lambda, \mathcal{R}_2)$ and $crs_3 \leftarrow \mathtt{setup}(1^\lambda, \mathcal{R}_3)$ for the zero knowledge proof system.
3. Set up the correlated randomness $(r_1, \ldots, r_n) \leftarrow \mathcal{D}^{\mathsf{bc}}_{\mathtt{corr}}$ and distribute r_i to party P_i for $i \in [n]$.

First round. Every party P_i proceeds as follows:

1. Let $\mathsf{msg}^1_i = \mathtt{frst\text{-}msg}_i(x_i, r_i)$ be P_i's first-round message in Π_{bc}.
2. Compute $(\mathtt{GC}_i, \vec{K}_i) \leftarrow \mathtt{garble}(1^\lambda, \mathtt{C}_i; R_i)$, where $\vec{K}_i = \{K^0_j, K^1_j\}_{j \in [L]}$.
3. For every index $l \in [k+1, \ldots, L]$ corresponding to a first-round message and $b \in \{0,1\}$, compute $(k^b_{i\to 1,l}, \ldots, k^b_{i\to n,l}) \leftarrow \mathtt{share}(K^b_l; R_i)$.
4. For every $j \in [n]$, let $\mathtt{k}_{i,j} \leftarrow \mathtt{keyagree}(\mathtt{pk}_j, \mathtt{sk}_i)$ be P_i's shared key with P_j. For every $l \in [k+1, \ldots, L]$, $b \in \{0,1\}$ and $j \in [n]$, compute $\overline{k}^b_{i\to j,l} \leftarrow \mathtt{enc}(\mathtt{k}_{i,j}, k^b_{i\to j,l}; R_i)$.
5. $c_{\mathtt{GC}_i}, o_{\mathtt{GC}_i} \leftarrow \mathtt{commit}(\mathtt{GC}_i)$.
6. Let $\mathsf{msg}_{i,\mathsf{GC}} = (c_{\mathtt{GC}_i}, \{\{\overline{k}^0_{i\to j,l}, \overline{k}^1_{i\to j,l}\}_{l\in[k+1,\ldots,L]}\}_{j\in[n]})$.
7. Let $\phi = (\{\mathtt{pk}_j\}_{j\in[n]}, c_{R,i}, \mathsf{msg}_{i,\mathsf{GC}})$, and $w = (\mathtt{C}_i, \mathtt{GC}_i, R_i, o_{\mathtt{GC}_i}, o_i, \mathtt{sk}_i)$. Compute $\pi_i \leftarrow \mathtt{prove}(crs_1, \phi, w)$.
8. Send to all parties the message $(\mathsf{msg}^1_i, \mathsf{msg}_{i,\mathsf{GC}}, \pi_i)$.

Second Round. Every party P_i proceeds as follows:

1. Computes labels for its own garbled circuit corresponding to its own input: Let $(\nu_{i,1}, \ldots, \nu_{i,k})$ correspond to the bits of P_i's input (x_i, r_i). For every $l \in [k]$, let $K_{i,l} = K^{\nu_{i,l}}_l$.
2. Computes shares of labels for all garbled circuits as follows.
   - Let $L_{i,bad} = \{\}$. Here, P_i will store shared keys with parties P_j who did not provide accepting π_j.
   - Let $L_{i,good} = \{\}$. Here, P_i will store shares of labels for $\mathtt{GC}_j$ for accepting π_j.
   - For every $j \in [n]$, let

$$(\mathsf{msg}^1_{j\to i}, \mathsf{msg}_{j,\mathsf{GC}}, \pi_j) =$$
$$(\mathsf{msg}^1_{j\to i}, (c_{\mathtt{GC}_j}, \{\{\overline{k}^0_{j\to j',l}, \overline{k}^1_{j\to j',l}\}_{l\in[k+1,\ldots,L]}\}_{j'\in[n]}), \pi_j)$$

   - Denote the concatenation of all the messages $\mathsf{msg}^1_{j\to i}$ as

$$(\nu_{i,k+1}, \ldots, \nu_{i,L}) = (\mathsf{msg}^1_{1\to i}, \ldots, \mathsf{msg}^1_{n\to i}) \in \{0,1\}^{n\cdot l}.$$

   - For $j \in [n]$,
     - Let $\mathtt{k}_{j,i} \leftarrow \mathtt{keyagree}(\mathtt{pk}_j, \mathtt{sk}_i)$.
     - Verify π_j.

    - ∗ It π_j does not verify, let $\pi_{i,j} \leftarrow \mathtt{prove}(crs_3, \phi_{i,j} = (\mathtt{pk}_i, \mathtt{pk}_j, \mathtt{k}_{j,i}), w_{i,j} = (\mathtt{sk}_i))$, and add $(\mathtt{k}_{j,i}, \pi_{i,j})$ to $L_{i,bad}$. This enables anyone to later extract all shares sent from P_j to P_i.
    - ∗ If π_j verifies,
      - · For $l \in [k+1, \ldots, L]$, let $k^l_{j\to i} \leftarrow \mathtt{dec}(\mathtt{k}_{j,i}, \overline{k}^{\nu_{i,l}}_{j\to i})$.
      - · Let $\phi_{i,j} = (\mathtt{pk}_i, \mathtt{pk}_j, \{\overline{k}^{\nu_{i,l}}_{j\to i}, k^l_{j\to i}\}_{l\in[k+1,\ldots,L]})$ and $w_{i,j} = \mathtt{sk}_i$. Let $\pi_{i,j} \leftarrow \mathtt{prove}(crs_2, \phi_{i,j}, w_{i,j})$.
      - · Add $(\mathsf{msg}^1_{j\to i}, k_{j\to i}, \pi_{i,j})$ to $L_{i,good}$.
3. Broadcast $(\mathtt{GC}_i, \mathsf{msg}_{i,\mathsf{GC}}, \pi_i, \{K_{i,l}\}_{l\in[k]}, L_{i,good}, L_{i,bad})$.

Output. Every party P_i proceeds as follows:

1. For every $j \in [n]$, let

   $$(\mathtt{GC}_i, \mathsf{msg}_{j,\mathsf{GC}}) =$$
   $$(c_{\mathtt{GC}_j}, \{\{\overline{k}^0_{j\to j',l}, \overline{k}^1_{j\to j',l}\}_{l\in[k+1,\ldots,L]}\}_{j'\in[n]}), \pi_j, \{K_{j,l}\}_{l\in[k]}, L_{j,good}, L_{j,bad})$$

   be the second-round message received from P_j.
2. For $j \in [n]$:
   - – If π_j does not verify, output $\mathtt{abort}_j$.
   - – Let good_j be the set of j' such that an element $(\mathsf{msg}^1_{j\to j'}, k_{j\to j'}, \pi_{j',j})$ is in $L_{j',good}$, and let bad_j be the set of j' such that an element $(\mathtt{k}_{j,j'}, \pi_{j',j})$ is in $L_{j',bad}$.
   - – Define msg^1_j as the message equal to at least $n-t$ of $\mathsf{msg}^1_{j\to j'}$ for $j' \in \mathsf{good}_j$. If such a message does not exist, output $\mathtt{abort}_j$. Remove all j' with messages that do not match msg^1_j from good_j.
3. Let $(\nu_{k+1}, \ldots, \nu_L) = (\mathsf{msg}^1_1, \ldots, \mathsf{msg}^1_n)$.
4. For $j \in [n]$:
   - – For $j' \in \mathsf{good}_j$, if $\pi_{j',j}$ does not verify, output $\mathtt{abort}_{j'}$.
   - – For $j' \in \mathsf{bad}_j$, if $\pi_{j',j}$ does not verify, output $\mathtt{abort}_{j'}$. Otherwise, for $l \in [k+1, \ldots, L]$, let $k_{j\to j',l} \leftarrow \mathtt{dec}(\mathtt{k}_{j,j'}, \overline{k}^{\nu_l}_{j\to j'})$.
   - – For $l \in [k+1, \ldots, L]$, compute $K_{j,l} = \mathtt{reconstruct}(\{k_{j\to j',l}\}_{j'\in\mathsf{good}_j\cup\mathsf{bad}_j})$. If the reconstruction fails output $\mathtt{abort}_j$.
   - – Evaluate the garbled circuit $\mathtt{GC}_j$ as $\mathsf{msg}^2_j = \mathtt{eval}(\mathtt{GC}_j, K_{j,1}, \ldots, K_{j,L})$. If the evaluation fails, $\mathtt{abort}_j$.
5. Output $y = \mathtt{output}_i(x_i, r_i, (\mathsf{msg}^1_1, \ldots, \mathsf{msg}^1_n), (\mathsf{msg}^2_1, \ldots, \mathsf{msg}^2_n))$.

Theorem 8 (P2P-BC, ID, $n > 3t$). *Let $\mathcal{F}$ be an efficiently computable n-party function and let $n > 3t$. Let $\Pi_{\mathtt{bc}}$ be a two broadcast-round protocol that securely computes $\mathcal{F}$ with identifiable abort with a black-box straight-line simulator. Assume that* $(\mathtt{garble}, \mathtt{eval}, \mathtt{simGC})$ *is a secure garbling scheme,* $(\mathtt{share}, \mathtt{reconstruct}, \mathtt{simshare})$ *is a secure secret sharing scheme with threshold* $\frac{2n}{3}+1$*,*

$(\texttt{enc}, \texttt{dec})$ *is a secure symmetric encryption scheme with key space* $\mathcal{D}_{\texttt{k}}$, $(\texttt{commit}, \texttt{open})$ *is a secure commitment scheme,* $(\texttt{keygen}, \texttt{keyagree})$ *is a secure non-interactive key exchange protocol with key distribution* $\mathcal{D}_{\texttt{k}}$, *and* $(\texttt{setup}, \texttt{prove}, \texttt{verify}, \texttt{simP}, \texttt{simP.Extract})$ *is a secure non-interactive zero-knowledge proof system. Then,* $\mathit{\Pi}_{\texttt{p2pbc}}$ *securely computes* $\mathcal{F}$ *with identifiable abort over two rounds, the first of which is over peer-to-peer channels, and the second of which is over a broadcast channel.*

Proof intuition. The protocol $\mathit{\Pi}_{\texttt{p2p-bc}}$ is a compiler, and should take as input any two-broadcast-round protocol $\mathit{\Pi}_{\texttt{bc}}$ that securely computes $\mathcal{F}$ with unanimous identifiable abort. Therefore the corresponding simulator $\mathcal{S}$, that interacts with adversary $\mathcal{A}$, relies on the a simulator $\mathcal{S}_{\texttt{bc}}$ for $\mathit{\Pi}_{\texttt{bc}}$. In particular, our simulator $\mathcal{S}$ runs $\mathcal{S}_{\texttt{bc}}$ internally, acting like a proxy for the messages of $\mathit{\Pi}_{\texttt{bc}}$: she forwards the malicious parties' $\mathit{\Pi}_{\texttt{bc}}$ messages received from $\mathcal{A}$ to $\mathcal{S}_{\texttt{bc}}$, and she uses the messages of $\mathcal{S}_{\texttt{bc}}$ to simulate the honest parties' messages (that she compiles into $\mathit{\Pi}_{\texttt{p2p-bc}}$ messages before forwarding to $\mathcal{A}$). Moreover, $\mathcal{S}$ simulates the trusted party's messages to $\mathcal{S}_{\texttt{bc}}$, forwarding the messages that she gots from $\mathcal{S}_{\texttt{bc}}$ to her own trusted party (i.e., abort messages and the inputs/outputs of the adversary). Note that $\mathcal{S}_{\texttt{bc}}$ extracts corrupt parties' inputs from the first round message of $\mathit{\Pi}_{\texttt{bc}}$ (if did not abort), because the simulator is straight-line and the MPC protocol only has two rounds.

Unfortunately, as already observed by Cohen *et al.* [CGZ20], this strategy requires some finesse in our setting, since in the first round of $\mathit{\Pi}_{\texttt{p2p-bc}}$ (which is over peer-to-peer channels) the adversary could send different messages to different honest parties. The challenges we face are (1) defining the adversary $\mathcal{A}_{\texttt{bc}}$ to which $\mathcal{S}_{\texttt{bc}}$ should correspond, and (2) choosing which of potentially several different first-round messages from $\mathcal{A}$ to forward to $\mathcal{S}_{\texttt{bc}}$. (These challenges are largely related; the way in which we select messages to forward to $\mathcal{S}_{\texttt{bc}}$ defines the adversary to which $\mathcal{S}_{\texttt{bc}}$ should correspond.)

We follow a simulation strategy that is based on the one used by Cohen *et al.* We design a class of adversaries, one for each honest parties, called receiver-specific adversaries $\{\mathcal{A}_h\}_{h\in\mathbb{H}}$. Roughly speaking, a receiver-specific adversary $\mathcal{A}_h$ is executing $\mathit{\Pi}_{\texttt{bc}}$ with the honest parties and internally runs $\mathcal{A}$. $\mathcal{A}_h$ uses the malicious parties' messages generated by $\mathcal{A}$: if $\mathcal{A}$ sends $\frac{n}{3}+1$ equal messages to the honest parties in the first round, $\mathcal{A}_h$ forwards that message to all honest parties; otherwise she forwards the message received by the honest party P_h. Note that in $\mathit{\Pi}_{\texttt{p2p-bc}}$ in the second case (i.e., when there are more then $\frac{n}{3}+1$ of inconsistent messages) the adversary $\mathcal{A}$ will not recover any garbled circuit labels corresponding to this message; so, $\mathcal{S}$ sends $\texttt{abort}$ to the trusted party (blaming the sender of the inconsistent message) and simulates garbled circuits that output dummy value.

The security of $\mathit{\Pi}_{\texttt{bc}}$ guarantees that there exists a simulator for this class of adversaries; the simulator $\mathcal{S}$ for $\mathit{\Pi}_{\texttt{p2p-bc}}$ will make use of it. $\mathcal{S}$ uses the simulator $\mathcal{S}_k$ for $\mathcal{A}_k$ (where k is the smallest index in $\mathbb{H}$) using the "standard" simulation strategy explained in the beginning. It is important to notice that the strategies of $\mathcal{S}$ for generating the first round message for $\mathcal{A}$ is exactly the strategy of $\mathcal{A}_k$.

Proof. Let $\mathbb{A}$ and $\mathbb{H}$ be, respectively, the set of corrupted parties and of honest parties.

We assume that $\mathcal{A}$ is deterministic and that the output of $\mathcal{A}$ consists of her entire view during the protocol, i.e., the auxiliary information, the input and correlated randomness of all corrupted parties, and the messages received by honest parties during the protocol.

Receiver-specific adversaries.

- $\mathcal{A}_k$ simulates the setup for $\mathcal{A}$. In particular $\mathcal{A}_k$ sets up a PKI and runs $(crs_1, td_1) \leftarrow \mathtt{setup}(1^\lambda, \mathcal{R}_1)$, $(crs_2, td_2) \leftarrow \mathtt{setup}(1^\lambda, \mathcal{R}_2)$, and $(crs_3, td_3) \leftarrow \mathtt{setup}(1^\lambda, \mathcal{R}_3)$.
- Upon receiving the first-broadcast-round message msg_h^1 from an honest party P_h in $\Pi_{\mathtt{bc}}$ $\mathcal{A}_k$ does the following steps:
  1. For every $l \in [k+1, \ldots, L]$ and $b \in \{0,1\}$, it computes $k^b_{h\to 1,l}, \ldots, k^b_{h\to n,l} \leftarrow \mathtt{share}(0^L)$.
  2. For every $j \in [n]$ let $\mathtt{k}_{h,j} \leftarrow \mathtt{keyagree}(\mathtt{pk}_j, \mathtt{sk}_h)$ be P_h's shared key with P_j.
  3. For every $l \in [k+1, \ldots, L]$, $j \in \mathbb{H}$ and $b \in \{0,1\}$ compute $\bar{k}^b_{h\to j,l} \leftarrow \mathtt{enc}(\mathtt{k}_{h,j}, 0^L)$.
  4. For every $l \in [k+1, \ldots, L]$, $j \in \mathbb{A}$ and $b \in \{0,1\}$ compute $\bar{k}^b_{h\to j,l} \leftarrow \mathtt{enc}(\mathtt{k}_{h,j}, k^b_{h\to j,l})$.
  5. $c_{\mathtt{GC}_h}, o_{\mathtt{GC}_h} \leftarrow \mathtt{commit}(0^g)$.
  6. Let $\mathsf{msg}_{h,\mathtt{GC}} = (c_{\mathtt{GC}_h}, \{\{\bar{k}^0_{h\to j,l}, \bar{k}^1_{h\to j,l}\}_{[k+1,\ldots L]}\}_{j\in\mathbb{A}})$.
  7. Let $\phi = (\{\mathtt{pk}_j\}_{j\in[n]}, c_{r,h}, \mathsf{msg}_{h,\mathtt{GC}})$ and run $\pi_h \leftarrow \mathtt{simP}(crs_1, td_1, \phi)$.

  Then $\mathcal{A}_k$ sends $\mathsf{msg}_h^1, \mathsf{msg}_{h,\mathtt{GC}}, \pi_h$ on behalf of every honest party P_h to every corrupted P_i over the point-to-point channel in $\Pi_{\mathtt{p2p-bc}}$. Let $\{(\mathsf{msg}^1_{j\to h}, \mathsf{msg}_{j,\mathtt{GC}}, \pi_j)\}_{j\in\mathbb{A}}$ be the set of messages received from $\mathcal{A}$. For every dishonest party P_j if there are there are at least $\frac{1}{3}+1$ of messages $\mathsf{msg}^1_{j\to i_1}, \ldots, \mathsf{msg}^1_{j\to i_n}$, with $i_i \in \mathbb{H}$ that are equal $\mathcal{A}_k$ broadcasts one of them in $\Pi_{\mathtt{bc}}$, otherwise $\mathcal{A}_k$ broadcasts $\mathsf{msg}^1_{j\to k}$ in $\Pi_{\mathtt{bc}}$.
- Upon receiving the first-broadcast-round message msg_h^2 from an honest party P_h in $\Pi_{\mathtt{bc}}$ $\mathcal{A}_k$ computes the following steps:
  1. Denote
  $$(\nu_{h,k+1}, \ldots, \nu_{h,L}) = (\hat{\mathsf{msg}}^1_1, \ldots, \hat{\mathsf{msg}}^1_n) \in \{0,1\}^{n\cdot l}.$$
  2. Let $\mathtt{C}_h$ be the circuit computing $\mathtt{snd\text{-}msg}_h$ run $(\mathtt{GC}_h, K_{h,1}, \ldots, K_{h,L}) \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}_h, \hat{\mathsf{msg}}^2_h)$.
  3. Let $L_{h,bad} = \{\}$. Here, P_h will store shared keys with parties P_j who did not provide accepting π_j. Let $L_{h,good} = \{\}$. Here, P_h will store shares of labels for $\mathtt{GC}_j$ for accepting π_j.
  4. Verify π_j.
     - If $\mathtt{simP.Extract}$ applied on π_j aborts, let $\pi_{h,j} \leftarrow \mathtt{prove}(crs_3, \phi_{h,j} = (\mathtt{pk}_h, \mathtt{pk}_j, \mathtt{k}_{j,h}), w_{h,j} = \mathtt{sk}_h)$, and add $(\mathtt{k}_{j,h}, \pi_{h,j})$ to $L_{h,bad}$.

- Else,
  - For $l \in [k+1, \ldots, L]$, let $k^l_{j\to h} \leftarrow \mathtt{dec}(\mathtt{k}_{j,h}, \overline{k}^{\nu_{h,l}}_{j\to h})$.
  - For $j \in \mathbb{A}$ let $\phi_{h,j} = (\mathtt{pk}_h, \mathtt{pk}_j, \{\overline{k}^{\nu_{h,l}}_{j\to h}, k^l_{j\to h}\}_{l\in[k+1,\ldots,L]})$ let $\pi_{h,j} \leftarrow \mathtt{prove}(crs_2, \phi_{h,j}, w_{h,j} = \mathtt{sk}_h)$.
  - For $j \in \mathbb{H}$ let $\phi_{h,j} = (\mathtt{pk}_h, \mathtt{pk}_j, \{\overline{k}^{\nu_{h,l}}_{j\to h}, k^l_{j\to h}\}_{l\in[k+1,\ldots,L]})$ run $\pi_{h,j} \leftarrow \mathtt{simP}(crs_2, \phi_{h,j})$.
  - Add $(\mathsf{msg}^1_{j\to h}, k_{j\to h}, \pi_{h,j})$ to $L_{h,good}$.

$\mathcal{A}_k$ sends $(\mathsf{msg}_{h,\mathsf{GC}}, \pi_h, \{K_{h,l}\}_{l\in[k]}, L_{h,good}, L_{h,bad})$ on behalf of honest party P_h to $\mathcal{A}$.
Let

$$(\mathsf{msg}_{j,\mathsf{GC}} = (\mathtt{GC}_j, \{\{\overline{k}^0_{j\to j',l}, \overline{k}^1_{j\to j',l}\}_{l\in[k+1,\ldots,L]}\}_{j'\in[n]}), \pi_j, \{K_{j,l}\}_{l\in[k]},$$

$$L_{j,good}, L_{j,bad})$$

be the second-round message received from corrupted party P_j for $j \in \mathbb{A}$.

– For $j \in [\mathbb{A}]$:
  - If $\mathtt{simP.Extract}$ applied on π_j aborts, $\mathcal{S}$ sets $\mathsf{msg}^2_j = \bot$.
  - Let good_j be the set of j' such that an element $(\mathsf{msg}^1_{j\to j'}, k_{j\to j'}, \pi_{j',j})$ is in $L_{j',good}$, and let bad_j be the set of j' such that an element $(\mathtt{k}_{j,j'}, \pi_{j',j})$ is in $L_{j',bad}$.
  - Define msg^1_j as the message equal to at least $n-t$ of $\mathsf{msg}^1_{j\to j'}$ for $j' \in \mathsf{good}_j$. If such a message does not exist, sets $\mathsf{msg}^2_j = \bot$. Remove all j' with messages that do not match msg^1_j from good_j.

– Let $(\nu_{k+1}, \ldots, \nu_L) = (\mathsf{msg}^1_1, \ldots, \mathsf{msg}^1_n)$.

– For $j \in [\mathbb{A}]$:
  - For $j' \in \mathsf{good}_j$, if $\mathtt{simP.Extract}$ applied on $\pi_{j',j}$ aborts, sets $\mathsf{msg}^2_j = \bot$.
  - For $j' \in \mathsf{bad}_j$, if $\mathtt{simP.Extract}$ applied on $\pi_{j',j}$ aborts, sets $\mathsf{msg}^2_j = \bot$.Otherwise, for $l \in [k+1, \ldots, L]$, let $k_{j\to j',l} \leftarrow \mathtt{dec}(\mathtt{k}_{j,j'}, \overline{k}^{\nu_l}_{j\to j'})$.
  - For $l \in [k+1, \ldots, L]$, compute $K_{j,l} = \mathtt{reconstruct}(\{k_{j\to j',l}\}_{j'\in \mathsf{good}_j \cup \mathsf{bad}_j})$. If the reconstruction fails sets $\mathsf{msg}^2_j = \bot$.
  - Evaluate the garbled circuit $\mathtt{GC}_j$ as $\mathsf{msg}^2_j = \mathtt{eval}(\mathtt{GC}_j, K_{j,1}, \ldots, K_{j,L})$. If the evaluation fails, sets $\mathsf{msg}^2_j = \bot$.

– Finally, $\mathcal{A}_k$ broadcasts the messages msg^2_j for every corrupted P_j in Π_{bc}, outputs whatever $\mathcal{A}$ outputs, and halts.

Simulator $\mathcal{S}$

By the security of $\Pi_{\mathtt{bc}}$, for every $k \in \mathbb{A}$ there exists a simulator $\mathcal{S}_k$ for the adversarial strategy $\mathcal{A}_k$ such that for every auxiliary information $\mathtt{aux}$ and input vector $x = (x_1, \ldots, x_n)$ it holds that ideal world and real world are computationally indistinguishable. Every simulator $\mathcal{S}_k$ starts by sending input values to her trusted party $\vec{x'_j} = \{x'_{i,j}\}_{i\in\mathbb{A}}$ Upon receiving the output value y, the simulator $\mathcal{S}_k$ sends a message $\mathtt{abort}_j$/continue (for some $j \in [n]$), and finally outputs the simulated view of the adversary, consisting of its input and the simulated messages of $\Pi_{\mathtt{bc}}$:

$$\hat{view}_j = \{\hat{\mathtt{aux}}^j, \{(x_i^j, r_i^j)\}_{i\in\mathbb{A}}, \hat{\mathsf{msg}}_1^{1,j}, \ldots, \hat{\mathsf{msg}}_n^{1,j}, \hat{\mathsf{msg}}_1^{2,j}, \ldots, \hat{\mathsf{msg}}_n^{2,j}\}.$$

Let $\mathcal{S}_{\mathtt{RS}}$ be the simulator $\mathcal{S}_j$ where k is the minimal index s.t. $k \in \mathbb{H}$. The simulator $\mathcal{S}$ start invoking $\mathcal{S}_{\mathtt{RS}}$ on her input and receiving back $\vec{x} = \{x_i\}_{i\in\mathbb{A}}$ or an $\mathtt{abort}_i$, for some $i \in [n]$. $\mathcal{S}$ simulates the interaction between $\mathcal{S}_{\mathtt{RS}}$ and the ideal functionality relaying on the trusted third part that computes $\mathcal{F}$. Specifically, $\mathcal{S}$ forwards the message that she received from $\mathcal{S}_{\mathtt{RS}}$ to the trusted third part and if $\mathcal{S}_{\mathtt{RS}}$ did not abort she receives back y. $\mathcal{S}$ forwards y to $\mathcal{S}_{\mathtt{RS}}$ which outputs the simulated view:

$$\hat{view}_{\mathtt{RS}} = \{\hat{\mathtt{aux}}, \{(x_i, r_i)\}_{i\in\mathbb{A}}, \hat{\mathsf{msg}}_1^1, \ldots, \hat{\mathsf{msg}}_n^1, \hat{\mathsf{msg}}_1^2, \ldots, \hat{\mathsf{msg}}_n^2\}.$$

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

Setup.

1. $\mathcal{S}$ sets up a PKI and runs $crs_1, td_1 \leftarrow \mathtt{setup}(1^\lambda)$, $crs_2, td_2 \leftarrow \mathtt{setup}(1^\lambda)$, and $crs_3, td_3 \leftarrow \mathtt{setup}(1^\lambda)$.
2. $\mathcal{S}$ invokes $\mathcal{A}$ on her inputs and the simulated correlated randomness for corrupted parties.

First round.

3. $\mathcal{S}$ on behalf of the honest party P_h for all $h \in \mathbb{H}$ computes the following steps:
   (a) For every $l \in [k+1, \ldots, L]$ and $b \in \{0,1\}$ computes $k^b_{h\to 1,l}, \ldots, k^b_{h\to n,l} \leftarrow \mathtt{share}(0^L)$.
   (b) For every $j \in [n]$ let $\mathtt{k}_{h,j} \leftarrow \mathtt{keyagree}(\mathtt{pk}_j, \mathtt{sk}_h)$ be P_h's shared key with P_j.
   (c) For every $l \in [k+1, \ldots, L]$, $j \in \mathbb{H}$ and $b \in \{0,1\}$ compute $\bar{k}^b_{h\to j,l} \leftarrow \mathtt{enc}(\mathtt{k}_{h,j}, 0^L)$.
   (d) For every $l \in [k+1, \ldots, L]$, $j \in \mathbb{A}$ and $b \in \{0,1\}$ compute $\bar{k}^b_{h\to j,l} \leftarrow \mathtt{enc}(\mathtt{k}_{h,j}, k^b_{h\to j,l})$.
   (e) $c_{\mathtt{GC}_h}, o_{\mathtt{GC}_h} \leftarrow \mathtt{commit}(0^g)$.
   (f) $\mathcal{S}$ sets $\mathsf{msg}_h^1 = \hat{\mathsf{msg}}_h^1$.
   (g) Let $\mathsf{msg}_{h,\mathsf{GC}} = (c_{\mathtt{GC}_h}, \{\{\bar{k}^0_{h\to j,l}, \bar{k}^1_{h\to j,l}\}_{[k+1,\ldots L]}\}_{j\in\mathbb{A}})$.
   (h) Let $\phi = (\{\mathtt{pk}_j\}_{j\in[n]}, c_{r,h}, \mathsf{msg}_{h,\mathsf{GC}})$ and run $\pi_h \leftarrow \mathtt{simP}(crs_1, td_1, \phi)$.

(i) Send $\mathsf{msg}^1_h, \mathsf{msg}_{h,\mathsf{GC}}, \pi_h$ to party P_i for $i \in \mathbb{A}$.

4. Let $\{(\mathsf{msg}^1_{i\to h}, \mathsf{msg}_{i,\mathsf{GC}}, \pi_i)\}_{i\in\mathbb{A}}$ be the set of messages received from $\mathcal{A}$.
Second round.
5. Let $L_{h,bad} = \{\}$. Here, P_h will store shared keys with parties P_j who did not provide accepting π_j. Let $L_{h,good} = \{\}$. Here, P_h will store shares of labels for $\mathtt{GC}_j$ for accepting π_j.
6. If for every $j \in \mathbb{A}$ there are at least $\frac{1}{3} + 1$ of messages $\mathsf{msg}^1_{j\to i_1}, \dots, \mathsf{msg}^1_{j\to i_n}$ equal to $\hat{\mathsf{msg}}^1_j$, with $i_i \in \mathbb{H}$, then $\mathcal{S}$ behaves as honest party P_h for $h \in \mathbb{H}$ and proceeds as follows; otherwise $\mathcal{S}$ executes Step 7:
(a) Denote
$$(\nu_{h,k+1}, \dots, \nu_{h,L}) = (\hat{\mathsf{msg}}^1_1, \dots, \hat{\mathsf{msg}}^1_n) \in \{0,1\}^{n\cdot l}.$$
(b) Let $\mathtt{C}_h$ be the circuit computing $\mathtt{snd\text{-}msg}_h$ run $(\mathtt{GC}_h, K_{h,1}, \dots, K_{h,L}) \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}_h, \hat{\mathsf{msg}}^2_h)$.
(c) For every $l \in [k+1, \dots, L]$ $\mathcal{S}$ runs $k^{\nu_{h,l}}_{h\to i_1,l}, \dots, k^{\nu_{h,l}}_{h\to i_{|\mathbb{H}|},l} \leftarrow \mathtt{simshare}(K_{h,l}, \tau, k^{\nu_{h,l}}_{h\to i_1,l}, \dots, k^{\nu_{h,l}}_{h\to i_{|\mathbb{A}|},l})$.
7. Let $j \in \mathbb{A}$ be the minimal index for which there are less then $\frac{1}{3} + 1$ of messages $\mathsf{msg}^1_{j\to i_1}, \dots, \mathsf{msg}^1_{j\to i_n}$ equal to $\hat{\mathsf{msg}}^1_j$, with $i_i \in \mathbb{H}$, then $\mathcal{S}$ behaves as honest party P_h for $h \in \mathbb{H}$ and proceeds as follows:
(a) $\mathcal{S}$ sends $\mathtt{abort}_j$ to the trusted third part.
(b) $\mathcal{S}$ runs $(\mathtt{GC}_h, K_{h,1}, \dots, K_{h,L}) \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}_h, 0^L)$.
(c) For every $g \in \mathbb{H}$ let $\mathsf{msg}^1_{h\to g} = \hat{\mathsf{msg}}^1_h$ and denote
$$(\nu_{h,1}, \dots, \nu_{h,L}) = (\mathsf{msg}^1_{1\to h}, \dots, \mathsf{msg}^1_{n\to h}) \in \{0,1\}^L.$$
8. Verify π_j.
   – If $\mathtt{simP.Extract}$ applied on π_j aborts, let $\pi_{h,j} \leftarrow \mathtt{prove}(crs_3, \phi_{h,j} = (\mathtt{pk}_h, \mathtt{pk}_j, \mathtt{k}_{j,h}), w_{h,j} = \mathtt{sk}_h)$, and add $(\mathtt{k}_{j,h}, \pi_{h,j})$ to $L_{h,bad}$.
   – Else,
     • For $l \in [k+1, \dots, L]$, let $k^l_{j\to h} \leftarrow \mathtt{dec}(\mathtt{k}_{j,h}, \overline{k}^{\nu_{h,l}}_{j\to h})$.
     • For $j \in \mathbb{A}$ let $\phi_{h,j} = (\mathtt{pk}_h, \mathtt{pk}_j, \{\overline{k}^{\nu_{h,l}}_{j\to h}, k^l_{j\to h}\}_{l\in[k+1,\dots,L]})$ let $\pi_{h,j} \leftarrow \mathtt{prove}(crs_2, \phi_{h,j}, w_{h,j} = \mathtt{sk}_h)$.
     • For $j \in \mathbb{H}$ let $\phi_{h,j} = (\mathtt{pk}_h, \mathtt{pk}_j, \{\overline{k}^{\nu_{h,l}}_{j\to h}, k^l_{j\to h}\}_{l\in[k+1,\dots,L]})$ run $\pi_{h,j} \leftarrow \mathtt{simP}(crs_2, \phi_{h,j})$.
     • Add $(\mathsf{msg}^1_{j\to h}, k_{j\to h}, \pi_{h,j})$ to $L_{h,good}$.
9. $\mathcal{S}$ $(\mathsf{msg}_{h,\mathsf{GC}}, \pi_h, \{K_{h,l}\}_{l\in[k]}, L_{h,good}, L_{h,bad})$ sends to $\mathcal{A}$.
Output.
10. For every $j \in [\mathbb{A}]$, let
$$(\mathsf{msg}_{j,\mathsf{GC}} = (\mathtt{GC}_j, \{\{\overline{k}^0_{j\to j',l}, \overline{k}^1_{j\to j',l}\}_{l\in[k+1,\dots,L]}\}_{j'\in[n]}), \pi_j, \{K_{j,l}\}_{l\in[k]}, L_{j,good}, L_{j,bad})$$
be the second-round message received from P_j. $\mathcal{S}$ checks:

(a) For $j \in [\mathbb{A}]$:
   i. If $\texttt{simP.Extract}$ applied on π_j aborts, $\mathcal{S}$ sends $\texttt{abort}_j$ to the trusted third part.
   ii. Let good_j be the set of j' such that an element $(\mathsf{msg}^1_{j \to j'}, k_{j \to j'}, \pi_{j',j})$ is in $L_{j',good}$, and let bad_j be the set of j' such that an element $(\mathtt{k}_{j,j'}, \pi_{j',j})$ is in $L_{j',bad}$.
   iii. Define msg^1_j as the message equal to at least $n - t$ of $\mathsf{msg}^1_{j \to j'}$ for $j' \in \mathsf{good}_j$. If such a message does not exist, sends $\texttt{abort}_j$ to the trusted third part. Remove all j' with messages that do not match msg^1_j from good_j.

(b) Let $(\nu_{k+1}, \ldots, \nu_L) = (\mathsf{msg}^1_1, \ldots, \mathsf{msg}^1_n)$.

(c) For $j \in [\mathbb{A}]$:
   i. For $j' \in \mathsf{good}_j$, if $\texttt{simP.Extract}$ applied on $\pi_{j',j}$ aborts, sends $\texttt{abort}_j$ to the trusted third part.
   ii. For $j' \in \mathsf{bad}_j$, if $\texttt{simP.Extract}$ applied on $\pi_{j',j}$ aborts, sends $\texttt{abort}_j$ to the trusted third part. Otherwise, for $l \in [k + 1, \ldots, L]$, let $k_{j \to j',l} \leftarrow \texttt{dec}(\mathtt{k}_{j,j'}, \overline{k}^{\nu_l}_{j \to j'})$.
   iii. For $l \in [k + 1, \ldots, L]$, compute $K_{j,l} = \texttt{reconstruct}(\{k_{j \to j',l}\}_{j' \in \mathsf{good}_j \cup \mathsf{bad}_j})$. If the reconstruction fails output $\texttt{abort}_j$.
   iv. Evaluate the garbled circuit $\texttt{GC}_j$ as $\mathsf{msg}^2_j = \texttt{eval}(\texttt{GC}_j, K_{j,1}, \ldots, K_{j,L})$. If the evaluation fails, sends $\texttt{abort}_j$ to the trusted third part.

If $\mathcal{S}$ did not abort sends $\texttt{continue}$ to the trusted third part.

11. $\mathcal{S}$ outputs the output of $\mathcal{A}$ and terminates.

We will now proceeds through a series oh hybrid experiments in oder to prove that the joint distribution of the output of $\mathcal{A}$ and the output of the honest parties in the ideal execution is computationally indistinguishable from the joint distribution of the output of $\mathcal{A}$ and the output of honest parties in a real protocol execution. The hybrid experiments are listed below. The output of the experiments is defined as the output of $\mathcal{A}$ and the output of the honest parties.

1. $\texttt{Expt}^0_{\mathbb{A},\mathcal{A},\mathit{\Pi}_{\texttt{p2p-bc}}}$: In this experiments, the simulator $\mathcal{S}_0$ has access to the internal state of the trusted party computing $\mathcal{F}$, therefore $\mathcal{S}_0$ chooses the output values of the honest parties. In the execution of $\mathit{\Pi}_{\texttt{p2p-bc}}$ the simulator is interacting with $\mathcal{A}$ on behalf of the honest parties. The output of this hybrid experiment is the output of the honest parties and the output of $\mathcal{A}$ in the execution of $\mathit{\Pi}_{\texttt{p2p-bc}}$ explained above. It follows trivially that the output of $\texttt{Expt}^0_{\mathbb{A},\mathcal{A},\mathit{\Pi}_{\texttt{p2p-bc}}}$ and the the output of the real world experiment are identically distributed.

2. $\mathtt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$: in this experiment $\mathtt{Expt}^0_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ is modified as follows. The simulator $\mathcal{S}_1$ start invoking $\mathcal{S}_{\mathtt{RS}}$ on her input and receiving back $\vec{x} = \{x_i\}_{i\in\mathbb{A}}$ or an $\mathtt{abort}_i$, for some $i \in [n]$. $\mathcal{S}_1$ simulates the interaction between $\mathcal{S}_{\mathtt{RS}}$ and the ideal functionality relaying on the trusted third part. Specifically, $\mathcal{S}_1$ forwards the message that she received from $\mathcal{S}_{\mathtt{RS}}$ to $\mathcal{F}$ and if $\mathcal{S}_{\mathtt{RS}}$ did not abort she receives back y. $\mathcal{S}_1$ forwards y to $\mathcal{S}_{\mathtt{RS}}$.
   $\mathcal{S}_1$ checks that more than 1/3 of consistent messages have been sent to the honest parties for each $j \in \mathbb{A}$. If this is not the case $\mathcal{S}_1$ outputs $\mathtt{abort}_j$ (where $j \in \mathbb{A}$ is the minimal index for which $\mathcal{A}$ sends more then 1/3 of inconsistent messages). $\mathcal{S}_1$ executes also the same checks that the ideal world simulator $\mathcal{S}$ (described above) in steps 10(c)iii and 10(c)iv does. If one of the checks fail $\mathcal{S}_1$ aborts identifying the cheater accordingly to the strategy of $\mathcal{S}$ in the correspondisg steps.

   Claim. $\mathtt{Expt}^0_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ and $\mathtt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ are computationally indistinguishable.

   Proof (Sketch). If $\mathcal{A}$ sends more then 1/3 of inconsistent messages of $\Pi_{\mathtt{bc}}$ for some malicious party in $\mathtt{Expt}^0_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ the honest parties output $\mathtt{abort}_j$ (where $j \in \mathbb{A}$ is the minimal index for which $\mathcal{A}$ sends more then 1/3 of inconsistent messages). Note that in this case the same check fails also in $\mathtt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$, therefore $\mathcal{S}_1$ sends $\mathtt{abort}_j$ to her trusted third part making sure that the honest parties also in $\mathtt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ output $\mathtt{abort}_j$. If the check above did not fail, then $\mathcal{A}$ recovers garble circuits and labels of the honest parties and therefore $\mathcal{A}$ gets to learn the output. At this point $\mathcal{A}$ could sends labels and garble circuits on behalf of dishonest parties. If the garble circuit evaluation or the the secret sharing reconstruction fails honest parties abort identifying the cheater j (where $j \in \mathbb{A}$ is the minimal index for which garble circuit evaluation or the the secret sharing reconstruction fails). In this case one of the checks in steps 10(c)iii and 10(c)iv fail and $\mathcal{S}_1$ in $\mathtt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ sends $\mathtt{abort}_j$ to $\mathcal{F}$. We conclude that honest parties aborts in $\mathtt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ only when the honest parties are aborting in $\mathtt{Expt}^0_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$. If all the checks above did not fail then it is possible to claim that $\mathtt{Expt}^0_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ and $\mathtt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ are computationally indistinguishable relying on the security of $\Pi_{\mathtt{bc}}$.

3. $\mathtt{Expt}^2_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$: in this experiment $\mathtt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ is modified as follows. In order to compute π_h in the first round $\mathcal{S}_2$ runs $\mathtt{simP}(crs_1, td_1, \phi)$, for $h \in \mathbb{H}$. Moreover $\mathcal{S}_2$ executes the step 8 of $\mathcal{S}$.

   Claim. $\mathtt{Expt}^2_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ and $\mathtt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ are computationally indistinguishable.

   Proof (Sketch). The proof proceeds via $|\mathbb{H}| + 1$ hybrids arguments: in the j-th hybrid experiment π_h of honest party P_h with $h \leq j$ are simulated as in $\mathtt{Expt}^2_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ and for $h > j$ are computed as $\mathtt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$. In order to claim that two neighboring hybrids are computationally indistinguishable we can rely on the simulation extractability property of the zero-knowledge proof system. Notice that the abort probability between two neighboring hybrids increases only of a negligible amount due to simulation extractability of the zero-knowledge proof system. The proof conclude observing that the 0-th hybrid corresponds to $\texttt{Expt}^1_{\mathbb{A},\mathcal{A},\mathit{\Pi}_{\texttt{p2p-bc}}}$ and the $|\mathbb{H}|$-th corresponds to $\texttt{Expt}^2_{\mathbb{A},\mathcal{A},\mathit{\Pi}_{\texttt{p2p-bc}}}$.

4. $\texttt{Expt}^3_{\mathbb{A},\mathcal{A},\mathit{\Pi}_{\texttt{p2p-bc}}}$: in this experiment $\texttt{Expt}^2_{\mathbb{A},\mathcal{A},\mathit{\Pi}_{\texttt{p2p-bc}}}$ is modified as follows. let $\phi_{h,j} = (\texttt{pk}_h, \texttt{pk}_j, \{\overline{k}^{\nu_{h,l}}_{j\to h}, k^l_{j\to h}\}_{l\in[k+1,\ldots,L]})$ in the second round $\mathcal{S}_3$ runs $\pi_{h,j} \leftarrow \texttt{simP}(crs_2, \phi_{h,j})$, where $\{\overline{k}^{\nu_{h,l}}_{j\to h}, k^l_{j\to h}\}_{l\in[k+1,\ldots,L]}$ are computed as in $\mathit{\Pi}_{\texttt{p2p-bc}}$. Moreover $\mathcal{S}_3$ executes the step, 10(c)i 10(c)ii of $\mathcal{S}$.

   Claim. $\texttt{Expt}^2_{\mathbb{A},\mathcal{A},\mathit{\Pi}_{\texttt{p2p-bc}}}$ and $\texttt{Expt}^3_{\mathbb{A},\mathcal{A},\mathit{\Pi}_{\texttt{p2p-bc}}}$ are computationally indistinguishable.

   Proof (Sketch). The proof proceeds in a similar way to the on described for Claim 3

5. $\texttt{Expt}^4_{\mathbb{A},\mathcal{A},\mathit{\Pi}_{\texttt{p2p-bc}}}$: in equal to experiment $\texttt{Expt}^3_{\mathbb{A},\mathcal{A},\mathit{\Pi}_{\texttt{p2p-bc}}}$ except that the simulator $\mathcal{S}_4$ for the honest party sends in the first round commitment of 0^g. In more detail, $\mathcal{S}_4$ computes $c_{\texttt{GC}_h}, o_{\texttt{GC}_h} \leftarrow \texttt{commit}(0^g)$ for all $h \in \mathbb{H}$.

   Claim. $\texttt{Expt}^4_{\mathbb{A},\mathcal{A},\mathit{\Pi}_{\texttt{p2p-bc}}}$ and $\texttt{Expt}^3_{\mathbb{A},\mathcal{A},\mathit{\Pi}_{\texttt{p2p-bc}}}$ are computationally indistinguishable.

   Proof (Sketch). The proof proceeds via $|\mathbb{H}|+1$ hybrids arguments: in the j-th hybrid experiment $c_{\texttt{GC}_h}$ of honest party P_h with $h \le j$ are commitment of 0^g as in $\texttt{Expt}^4_{\mathbb{A},\mathcal{A},\mathit{\Pi}_{\texttt{p2p-bc}}}$ and for $h > j$ are computed as $\texttt{Expt}^3_{\mathbb{A},\mathcal{A},\mathit{\Pi}_{\texttt{p2p-bc}}}$. In order to claim that two neighboring hybrids are computationally indistinguishable we can rely on the hiding of the commitment scheme. The proof conclude observing that the 0-th hybrid corresponds to $\texttt{Expt}^4_{\mathbb{A},\mathcal{A},\mathit{\Pi}_{\texttt{p2p-bc}}}$ and the $|\mathbb{H}|$-th corresponds to $\texttt{Expt}^3_{\mathbb{A},\mathcal{A},\mathit{\Pi}_{\texttt{p2p-bc}}}$.

6. $\texttt{Expt}^5_{\mathbb{A},\mathcal{A},\mathit{\Pi}_{\texttt{p2p-bc}}}$: in equal to experiment $\texttt{Expt}^4_{\mathbb{A},\mathcal{A},\mathit{\Pi}_{\texttt{p2p-bc}}}$ except that the simulator $\mathcal{S}_5$ sends in the first round encryptions of 0^L to all the honest parties. In more detail, for every $l \in L$, $h, j \in \mathbb{H}$ and $b \in \{0,1\}$ $\mathcal{S}_5$ computes $\bar{k}^b_{h\to j,l} \leftarrow \texttt{enc}(\texttt{k}_{h,j}, 0^L)$.

   Proof (Sketch). The proof proceeds via $2(|\mathbb{H}|^2 l + 1)$ hybrids arguments:
   For each $h, h' \in \mathbb{H}$, $l \in [L]$, $b \in \{0,1\}$, we define hybrid experiment $\texttt{Expt}_{h\to h',l,b}$. In each subsequent experiment $\texttt{Expt}_{h\to h',l,b}$, the encryption $\bar{k}^b_{h\to h',l'}$ becomes an encryption of 0^L. In order to claim that two neighboring hybrids are computationally indistinguishable we can rely on security of the encryption scheme. The proof conclude observing that the 0-th hybrid corresponds to $\texttt{Expt}^4_{\mathbb{A},\mathcal{A},\mathit{\Pi}_{\texttt{bc-p2p}}}$ and the last hybrid corresponds to $\texttt{Expt}^5_{\mathbb{A},\mathcal{A},\mathit{\Pi}_{\texttt{bc-p2p}}}$.

7. $\mathtt{Expt}^{6}_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$: this experiment proceeds as the experiment $\mathtt{Expt}^{5}_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ except that the shares are computed differently. In more details, the simulator $\mathcal{S}_6$ computes the shares on behalf of the honest parties according to step 3a of the ideal world simulator $\mathcal{S}$ (described above).
   Let $(\nu_{h,1},\ldots,\nu_{h,L}) = (\mathsf{msg}^1_{1\to h},\ldots,\mathsf{msg}^1_{n\to h})$ and $\{m_{j\to h}\}_{j\in[n]}$ be the messages received by the honest party P_h in the first round of the execution of $\Pi_{\mathtt{p2p-bc}}$. If $\mathcal{A}$ does not send more then 1/3 of inconsistent messages of $\Pi_{\mathtt{bc}}$ to the honest parties for any corrupted party $\mathcal{S}_6$ computes for every $l \in [L]$ for all $h \in \mathbb{H}$ $\mathcal{S}$ $k^{\nu_{h,l}}_{h\to i_1,l},\ldots,k^{\nu_{h,l}}_{h\to i_{|\mathbb{H}|},l} \leftarrow \mathtt{simshare}(K_{h,l},k^{\nu_{h,l}}_{h\to i_1,l},\ldots,k^{\nu_{h,l}}_{h\to i_{|\mathbb{A}|},l})$.

   Claim. $\mathtt{Expt}^{5}_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ and $\mathtt{Expt}^{6}_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ are perfect indistinguishable.

   Proof. We can rely on the security of the secret sharing scheme to claim that $\mathtt{Expt}^{5}_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ and $\mathtt{Expt}^{6}_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ are perfect indistinguishable. Note that if $\mathcal{A}$ sends more then 1/3 of inconsistent messages of $\Pi_{\mathtt{bc}}$ in $\mathtt{Expt}^{5}_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ $\mathcal{A}$ is not able to recover any of the honest parties secrets (that are the labels for their garble circuit) since the reconstruction threshold is $t_{ss} = \frac{2}{3}n + 1$.

8. $\mathtt{Expt}^{7}_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$: this experiment proceeds as the experiment $\mathtt{Expt}^{6}_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ except that the garbled circuits regarding the honest parties are computed using the simulated procedure $\mathtt{simGC}$. In more detail, $\mathcal{S}_7$ executes, for all $h \in \mathbb{H}$, $(\mathtt{GC}_h, K_{h,1},\ldots,K_{h,L}) \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}_{h,x_h,r_h}, \mathsf{msg}^2_h)$, where msg^2_h is the message computed by P_h in the execution of $\Pi_{\mathtt{p2p-bc}}$ and $\mathtt{C}_{h,x_h,r_h}$ is the circuit $\mathtt{snd\text{-}msg}_h$ with hard-wired input x_h and randomness r_h.

   Claim. $\mathtt{Expt}^{7}_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ and $\mathtt{Expt}^{6}_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ are computationally indistinguishable.

   Proof. The proof proceeds via $|\mathbb{H}|+1$ hybrids arguments: in the j-th hybrid experiment the garble circuit of honest party P_h with $h \leq j$ are simulated as in $\mathtt{Expt}^{7}_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ and for $h > j$ are computed as in $\mathtt{Expt}^{6}_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$. In order to claim that two neighboring hybrids are computationally indistinguishable we can rely on security of garbling scheme. The proof conclude observing that the 0-th hybrid corresponds to $\mathtt{Expt}^{6}_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ and the $|\mathbb{H}|$-th corresponds to $\mathtt{Expt}^{7}_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$.

9. $\mathtt{Expt}^{8}_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$: in this experiment $\mathtt{Expt}^{7}_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ is modified as follows. Let $\mathtt{C}_h$ be the circuit $\mathtt{snd\text{-}msg}_h$ with hard-wired input and randomness set to 0. If $\mathcal{A}$ sends more then 1/3 of inconsistent messages of $\Pi_{\mathtt{bc}}$ for some malicious party $\mathcal{S}_8$ on behalf of honest party P_h (for all $h \in \mathbb{H}$) executes $(\mathtt{GC}_h, K_{h,1},\ldots,K_{h,L}) \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}_h, 0^L)$ (i.e. she garbles the circuit on a dummy output); otherwise she executes $(\mathtt{GC}_h, K_{h,1},\ldots,K_{h,L}) \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}_h, \mathsf{msg}^2_h)$, where msg^2_h is the message computed by P_h in the execution of $\Pi_{\mathtt{p2p-bc}}$.

   Claim. $\mathtt{Expt}^{8}_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ and $\mathtt{Expt}^{7}_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ are computationally indistinguishable.

Proof. This proceeds via hybrid experiments similar as in the proof of Claim 8, the only extra observation is that $\mathcal{A}$ does not learn the labels for the garble scheme if the $\mathcal{A}$ sends more then 1/3 of inconsistent messages of $\Pi_{\mathtt{bc}}$ both in $\mathtt{Expt}^8_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ that in $\mathtt{Expt}^7_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$.

10. $\mathtt{Expt}^9_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$: this experiment proceeds as the experiment $\mathtt{Expt}^8_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ except that instead of computing the messages of $\Pi_{\mathtt{bc}}$ relaying on honest parties inputs, the simulator $\mathcal{S}_9$ uses the messages given in output by $\mathcal{S}_{\mathtt{RS}}$. More in details, in the first round for all $h \in \mathbb{H}$ $\mathcal{S}_9$ sends $\hat{\mathsf{msg}}^1_h$, and in the second round computes $(\mathtt{GC}_h, K_{h,1}, \ldots, K_{h,L}) \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}_h, \hat{\mathsf{msg}}^2_h)$ (if the check in Step 6 does not fail).

    Claim. $\mathtt{Expt}^9_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ and $\mathtt{Expt}^8_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ are computationally indistinguishable.

    Proof. In $\mathtt{Expt}^9_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ the adversary learns the second messages of $\Pi_{\mathtt{bc}}$ w.r.t. the honest parties only when the honest parties agree on a same first round message msg^1_i of $\Pi_{\mathtt{bc}}$ from dishonest party P_i, that is when P_i sends more than 1/3 of consistent messages to the honest parties for the first round of $\Pi_{\mathtt{bc}}$. Therefore the indistinguishability between $\mathtt{Expt}^9_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ and $\mathtt{Expt}^8_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ follows from the security of $\Pi_{\mathtt{bc}}$. More in details, any distinguisher between $\mathtt{Expt}^9_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ and $\mathtt{Expt}^8_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ can be used to distinguish between $\mathcal{S}_{\mathtt{RS}}$ and real execution with the receiver-specific adversary.
    As observed in [CGZ20], the proof crucially rely on the ability oif the simulator $\mathcal{S}_{\mathtt{RS}}$ to extract the adversary's input from her first round of $\Pi_{\mathtt{bc}}$, and this is implied from the assumption that $\mathcal{S}_{\mathtt{RS}}$ is strait-line and black-box. Indeed, these properties for two rounds MPC guarantee that $\mathcal{S}_{\mathtt{RS}}$ will extract the adversary's input from her first round of $\Pi_{\mathtt{bc}}$.

The proof ends observing that in $\mathtt{Expt}^9_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ $\mathcal{S}_9$ does not need anymore to have access to the internal state of the trusted third part that computes $\mathcal{F}$ and therefore $\mathtt{Expt}^9_{\mathbb{A},\mathcal{A},\Pi_{\mathtt{p2p-bc}}}$ and the ideal world experiment are identically distributed.

P2P-BC Identifiable Abort with $n > 2t$ Next, we argue that the techniques we use to build $\Pi_{\mathtt{p2pbc}}$ do not apply when $n \leq 3t$. Let t_{MPC} denote the threshold of the secure computation protocol, and let t_{SS} denote the threshold of the secret sharing scheme we use (where $t_{SS}+1$ parties can reconstruct the secret, but t_{SS} cannot). We cannot achieve identifiable abort with the protocol structure of Cohen *et al.* for $t_{MPC} \geq \frac{n}{3}$, for the following reason.

In order to achieve identifiable abort, honest parties must be able to reconstruct garbled circuit labels without the help of malicious parties in the second round. Malicious parties could claim to receive a different first-round message from the underlying $\Pi_{\mathtt{bc}}$, and thus forward incorrect label shares. Honest parties will not know whether to blame those parties or the message sender, and since they are unable to identify a cheater, they must be able to reconstruct garbled circuit labels and complete the computation. So, we need

$$t_{SS} < n - t_{MPC}.$$

On the other hand, no malicious party should be able to reconstruct more than one label for the same input wire in an honest party's garbled second message function. Since the first message is sent over peer-to-peer channels, a malicious party can always send 0 to half the honest parties (getting their shares of the 0 label for a given wire), and 1 to the other half (getting their shares of the 1 label for the same wire). Malicious parties hold shares of both wires; so, it's important that they be unable to combine their shares with only half the honest parties' shares to reconstruct a label. We get

$$t_{SS} \geq t_{MPC} + \frac{n - t_{MPC}}{2} = \frac{n + t_{MPC}}{2}.$$

t_{MPC} (satisfying both those constraints) is maximized at $\lceil \frac{n}{3} - 1 \rceil$.

While we are unable to use the techniques of Cohen *et al.* for $3t \geq n > 2t$, we conjecture that we could use less efficient techniques such as obfuscation, and stronger setup.

Conjecture 1 (P2P-BC, ID, $n > 2t$). Given (a) obfuscation and (b) correlated randomness, we can achieve secure computation with identifiable abort in two rounds only the second of which is over a broadcast channel, for $n > 2t$.

We informally sketch a construction achieving secure computation with identifiable abort as per Conj 1. We note that this construction is incredibly impractical; we sketch it here simply to demonstrate feasibility, or, rather, the *in*feasibility of a negative result. We believe that its security can be based on indistinguishability obfuscation.

The construction requires a PKI, and correlated randomness in the form of an obfuscated program which is hardcoded with (a) the function being computed, (b) all parties' public keys, and (c) a secret decryption key. Parties send all their peers a signed encryption of their input in the first round; in the second round, they broadcast signed echos of those encryptions. They then use the echos as inputs to the obfuscated program, which gives them the output. The echos provide a *verifiable broadcast* of the encrypted inputs, in the sense that after the second round, each party holds proof that sufficiently many other parties saw the same encrypted inputs. This proof is checked by the program before it produces an output, and serves to prevent the adversary from recomputing the output with different inputs.

In more detail, consider the function $\mathtt{output}_{f,\mathtt{sk},\mathtt{pk}_1,\ldots,\mathtt{pk}_n}$ (described in Figure 1), in which a secret decryption key $\mathtt{sk}$ (the public encryption key $\mathtt{pk}$ corresponding to which is known to all parties) is hard-coded, along with the function f, and all parties' signature verification keys $\mathtt{pk}_1, \ldots, \mathtt{pk}_n$. The obfuscation of $\mathtt{output}$, as well as the public encryption key $\mathtt{pk}$, is available in a CRS.

Each party P_i encrypts its input x_i to the CRS as $c_i \leftarrow \texttt{enc}(\texttt{pk}, x_i)$, and signs that ciphertext as $\sigma_i \leftarrow \texttt{sign}(\texttt{sk}_i, c_i)$, and sends all parties (c_i, σ_i) in the first round over peer-to-peer channels. Each party then concatenates all of the messages it got in the first round as $m_i = \{(c_j, \sigma_j)\}_{j \in \{1,\dots,n\} \setminus \{i\}}$, and signs this concatenation as $\sigma_i' \leftarrow \texttt{sign}(\texttt{sk}_i, m_i)$. P_i then broadcasts (m_i, σ_i') in the second round.

If a corrupt party does not broadcast anything in the second round (or provides a bad signature), it will be caught by the first for-loop of `output`. If a corrupt party sends different ciphertexts to different honest parties in the first round, it will be caught by the first if-statement of the second for-loop. If a corrupt party does not provide at least one honest party with its ciphertext, it will be caught by the second if-statement of that for-loop.

```
$\texttt{output}_{f,\texttt{sk},\texttt{pk}_1,\dots,\texttt{pk}_n}((m_1, \sigma_1'), \dots, (m_n, \sigma_n'))$ :
for $i \in [1, \dots, n]$ do
    if $\texttt{verify}(\texttt{pk}_i, m_i, \sigma_i') = \texttt{reject}$ then
        return $\texttt{abort}_i$
    end if
    Parse $m_i = \{(c_{j,i}, \sigma_{j,i})\}_{j \in \{1,\dots,n\} \setminus \{i\}}$
end for
Let $\mathsf{d}_{j,i} = \texttt{verify}(\texttt{pk}_j, c_{j,i}, \sigma_{j,i})$.
for $j \in [1, \dots, n]$ do
    if $\exists i, i'$ s.t. $c_{j,i} \neq c_{j,i'}$ and $\mathsf{d}_{j,i} = \mathsf{d}_{j,i'} = \texttt{accept}$ then
        return $\texttt{abort}_j$
    end if
    if $\nexists i_1 \neq \dots \neq i_{t+1}$ s.t. $c_{j,i_1} = \dots = c_{j,i_{t+1}}$ and $\mathsf{d}_{j,i_1} = \dots = \mathsf{d}_{j,i_{t+1}} = \texttt{accept}$ then
        return $\texttt{abort}_j$
    else
        $x_j \leftarrow \texttt{dec}(\texttt{sk}, c_{j,i_1})$
    end if
end for
return $f(x_1, \dots, x_n)$
```

Fig. 1: Function or Circuit to Obtain Computation Output

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, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 395–424. Springer, Heidelberg, August 2018.

ACGJ19. Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Two round information-theoretic MPC with malicious security. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 532–561. Springer, Heidelberg, May 2019.

BGI$^+$01. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001.

BGW88. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988.

BHR12. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012*, pages 784–796. ACM Press, October 2012.

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, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 500–532. Springer, Heidelberg, April / May 2018.

CCD88. David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th ACM STOC*, pages 11–19. ACM Press, May 1988.

CD01. Ronald Cramer and Ivan Damgård. Secure distributed linear algebra in a constant number of rounds. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 119–136. Springer, Heidelberg, August 2001.

CDN15. Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing.* Cambridge University Press, 2015.

CGZ20. Ran Cohen, Juan A. Garay, and Vassilis Zikas. Broadcast-optimal two-round MPC. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 828–858. Springer, Heidelberg, May 2020.

Cle86. Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *18th ACM STOC*, pages 364–369. ACM Press, May 1986.

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

FGMv02. Matthias Fitzi, Nicolas Gisin, Ueli M. Maurer, and Oliver von Rotz. Unconditional byzantine agreement and multi-party computation secure against dishonest minorities from scratch. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 482–501. Springer, Heidelberg, April / May 2002.

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.

GIKR01. Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. The round complexity of verifiable secret sharing and secure multicast. In *33rd ACM STOC*, pages 580–589. ACM Press, July 2001.

GIKR02. Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. On 2-round secure multiparty computation. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 178–193. Springer, Heidelberg, August 2002.

GLS15. S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 63–82. Springer, Heidelberg, August 2015.

GM17. Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In Jonathan Katz and

Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 581–612. Springer, Heidelberg, August 2017.

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 ACM STOC*, pages 218–229. ACM Press, May 1987.

GS18. Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 468–499. Springer, Heidelberg, April / May 2018.

IK02. Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan Eidenbenz, and Ricardo Conejo, editors, *ICALP 2002*, volume 2380 of *LNCS*, pages 244–256. Springer, Heidelberg, July 2002.

IKKP15. Yuval Ishai, Ranjit Kumaresan, Eyal Kushilevitz, and Anat Paskin-Cherniavsky. Secure computation with minimal interaction, revisited. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 359–378. Springer, Heidelberg, August 2015.

IKP10. Yuval Ishai, Eyal Kushilevitz, and Anat Paskin. Secure multiparty computation with minimal interaction. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 577–594. Springer, Heidelberg, August 2010.

KO04. Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 335–354. Springer, Heidelberg, August 2004.

Lin01. Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 171–189. Springer, Heidelberg, August 2001.

Ped92. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.

PR18. Arpita Patra and Divya Ravi. On the exact round complexity of secure three-party computation. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 425–458. Springer, Heidelberg, August 2018.

Yao82. Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.

Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

A Building Blocks

In this section we define the building blocks necessary for our protocols.

Symmetric Key Encryption

Definition 9. *A Symmetric-Key Encryption scheme is a tuple of efficient algorithms* $\texttt{ENC} = (\texttt{keygen}, \texttt{enc}, \texttt{dec})$ *defined as follows.*

> $\texttt{keygen}(1^\lambda) \to \texttt{sk}$**:** *The probabilistic algorithm* keygen *takes as input the security parameter* $\lambda \in \mathbb{N}$*, and outputs a secret key* sk*.*
> $\texttt{enc}(\texttt{sk}, m; r) \to c$**:** *The probabilistic algorithm* enc *takes as input the secret key* pk*, a message* $m \in \mathcal{M}$*, and implicit randomness* $\rho \in \mathcal{R}$*, and outputs a ciphertext* $c = \texttt{enc}(\texttt{sk}, m; r)$*. The set of all ciphertexts is denoted by* $\mathcal{C}$*.*
> $\texttt{dec}(\texttt{sk}, c) \to m$**:** *The deterministic algorithm* dec *takes as input the secret key* sk *and a ciphertext* $c \in \mathcal{C}$ *and outputs* $m = \texttt{dec}(\texttt{sk}, c)$ *which is either equal to some message* $m \in \mathcal{M}$ *or to an error symbol* $\perp$*.*

We require the following properties of a symmetric encryption scheme:

Correctness. We say that ENC *satisfies correctness if for all* $\texttt{sk} \leftarrow \texttt{keygen}(1^\lambda)$ *there exists a negligible function* $\nu : \mathbb{N} \to [0, 1]$ *such that that* $\Pr[\texttt{dec}(\texttt{sk}, \texttt{enc}(\texttt{sk}, m)) = m] \geq 1 - \nu(\lambda)$ *(where the randomness is taken over the internal coin tosses of algorithm* enc*).*

CPA security. We say that ENC *is CPA-secure if for all PPT adversaries* $\mathcal{A}$ *the following quantity is negligible:*

$$\Pr\left[b' = b : \begin{array}{c} b' \leftarrow \mathcal{A}(c); c \leftarrow \texttt{enc}(\texttt{sk}, m_b); b \leftarrow \{0, 1\} \\ (m_0, m_1) \leftarrow \mathcal{A}(1^\lambda); \texttt{sk} \leftarrow \texttt{keygen}(1^\lambda) \end{array}\right].$$

Public Key Encryption

Definition 10. *A Public-Key Encryption (PKE) scheme is a tuple of efficient algorithms* $\texttt{PKE} = (\texttt{keygen}, \texttt{enc}, \texttt{dec})$ *defined as follows.*

> $\texttt{keygen}(1^\lambda) \to (\texttt{pk}, \texttt{sk})$**:** *The probabilistic algorithm* keygen *takes as input the security parameter* $\lambda \in \mathbb{N}$*, and outputs a public/secret key pair* $(\texttt{pk}, \texttt{sk})$*.*
> $\texttt{enc}(\texttt{pk}, m; r) \to c$**:** *The probabilistic algorithm* enc *takes as input the public key* pk*, a message* $m \in \mathcal{M}$*, and implicit randomness* $\rho \in \mathcal{R}$*, and outputs a ciphertext* $c = \texttt{enc}(\texttt{pk}, m; r)$*. The set of all ciphertexts is denoted by* $\mathcal{C}$*.*
> $\texttt{dec}(\texttt{sk}, c) \to m$**:** *The deterministic algorithm* dec *takes as input the secret key* sk *and a ciphertext* $c \in \mathcal{C}$ *and outputs* $m = \texttt{dec}(\texttt{sk}, c)$ *which is either equal to some message* $m \in \mathcal{M}$ *or to an error symbol* $\perp$*.*

We require the following properties of a PKE scheme:

Correctness. A PKE scheme meets the correctness property if the decryption of a ciphertext encrypting a given plaintext yields the plaintext. We say that PKE *satisfies correctness if for all* $(\texttt{pk}, \texttt{sk}) \leftarrow \texttt{keygen}(1^\lambda)$ *there exists a negligible function* $\nu : \mathbb{N} \to [0, 1]$ *such that that* $\Pr[\texttt{dec}(\texttt{sk}, \texttt{enc}(\texttt{pk}, m)) = m] \geq 1 - \nu(\lambda)$ *(where the randomness is taken over the internal coin tosses of algorithm* enc*).*

CPA security. The standard security notion for PKE schemes goes under the name of security against chosen-plaintext attacks (CPA), and informally states that no efficient adversary given the public key can distinguish the encryption of two (possibly known) messages. Let $\texttt{PKE} = (\texttt{keygen}, \texttt{enc}, \texttt{dec})$ *be a PKE scheme. We say that* $\texttt{PKE}$ *is CPA-secure if for all PPT adversaries* $\mathcal{A}$ *the following quantity is negligible:*

$$\Pr\left[b' = b : \begin{array}{l} b' \leftarrow \mathcal{A}(\texttt{pk}, c); c \leftarrow \texttt{enc}(\texttt{pk}, m_b); b \leftarrow \{0,1\} \\ (m_0, m_1) \leftarrow \mathcal{A}(\texttt{pk}); (\texttt{pk}, \texttt{sk}) \leftarrow \texttt{keygen}(1^\lambda) \end{array}\right].$$

Non-Interactive Zero-Knowledge Arguments of Knowledge We take this definition from Groth and Maller [GM17].

Definition 11 (Non-Interactive Zero-Knowledge Arguments of Knowledge). *A non-interactive zero-knowledge argument of knowledge scheme* $\texttt{NIZK}$ *consists of the following algorithms:*

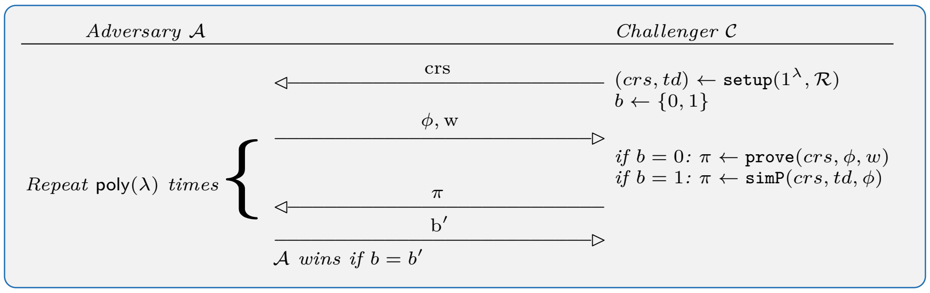
$\texttt{setup}(1^\lambda, \mathcal{R}) \to (crs, td)$**:** *Upon input the security parameter, sets up the global common reference string crs and the trapdoor td for the* $\texttt{NIZK}$ *system.*
$\texttt{prove}(crs, \phi, w) \to \pi$**:** *Upon input the common reference string crs for a relation* $\mathcal{R}$*, a statement* ϕ *and a witness* w*, returns a proof* π *that* $(\phi, w) \in \mathcal{R}$*.*
$\texttt{verify}(crs, \phi, \pi) \to \texttt{accept/reject}$**:** *Upon input the common reference string crs for a relation* $\mathcal{R}$*, a statement* ϕ *and a proof* π*, verifies whether* π *proves the existence of a witness* w *such that* $(\phi, w) \in \mathcal{R}$*.*
$\texttt{simP}(crs, td, \phi) \to \pi$**:** *Upon input the common reference string crs for a relation* $\mathcal{R}$*, the trapdoor td and a statement* ϕ*, simulates a proof of the existence of a witness* w *such that* $(\phi, w) \in \mathcal{R}$*.*

We require the following properties of a $\texttt{NIZK}$ *scheme:*

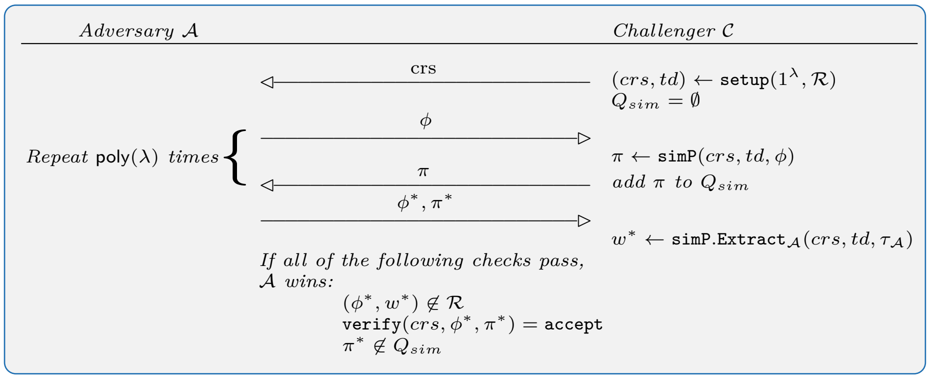
Completeness. For any $(\phi, w) \in \mathcal{R}$ *we have that*

$$\Pr\left[\texttt{verify}(\phi, \pi) = 1 \,\middle|\, \begin{array}{l} (crs, td) \leftarrow \texttt{setup}(1^\lambda, \mathcal{R}) \\ \pi \leftarrow \texttt{prove}(\phi, w) \end{array}\right] \geq 1 - negl(\lambda).$$

Zero Knowledge. We say that $\texttt{NIZK}$ *has zero-knowledge if for all PPT adversaries* $\mathcal{A}$ *and sufficiently large* λ*, there exists a negligible function* $negl(\lambda)$ *such that* $|\Pr[\mathcal{A} \text{ wins}] - \frac{1}{2}| \leq negl(\lambda)$ *in the following experiment:*


Adversary $\mathcal{A}$
Challenger $\mathcal{C}$
crs
$(crs, td) \leftarrow \texttt{setup}(1^\lambda, \mathcal{R})$
$b \leftarrow \{0,1\}$
ϕ, w
Repeat $\mathsf{poly}(\lambda)$ times
if $b = 0$: $\pi \leftarrow \texttt{prove}(crs, \phi, w)$
if $b = 1$: $\pi \leftarrow \texttt{simP}(crs, td, \phi)$
π
b'
$\mathcal{A}$ wins if $b = b'$

Simulation Extractability. We say that `NIZK` *is simulation extractable if for all PPT adversaries* $\mathcal{A}$ *and sufficiently large* λ*, there exists an extraction algorithm* $\texttt{simP.Extract}_{\mathcal{A}}$ *and a negligible function* $negl(\lambda)$ *such that* $\Pr[\mathcal{A}\ wins\ ] \leq negl(\lambda)$ *in the following experiment:*


Adversary $\mathcal{A}$
Challenger $\mathcal{C}$
crs
$(crs, td) \leftarrow \texttt{setup}(1^{\lambda}, \mathcal{R})$
$Q_{sim} = \emptyset$
Repeat $\mathsf{poly}(\lambda)$ times
ϕ
$\pi \leftarrow \texttt{simP}(crs, td, \phi)$
π
add π to Q_{sim}
ϕ^*, π^*
$w^* \leftarrow \texttt{simP.Extract}_{\mathcal{A}}(crs, td, \tau_{\mathcal{A}})$
If all of the following checks pass, $\mathcal{A}$ wins:
$(\phi^*, w^*) \notin \mathcal{R}$
$\texttt{verify}(crs, \phi^*, \pi^*) = \texttt{accept}$
$\pi^* \notin Q_{sim}$

Commitment Scheme A commitment scheme allows a party to commit to a value while keeping it hidden from the others, and later it allows the same party to reveal the committed value with the guarantee that the commitment is binding [Ped92].

Definition 12. *A commitment scheme consists of the following algorithms:*

$\mathsf{com}(\mathsf{msg}) \rightarrow (decom, com)$**:** *The commitment algorithm takes as input a message* $\mathsf{msg} \in \{0,1\}^{\lambda}$ *and outputs a decommitment value* $decom \in \{0,1\}^{\lambda}$ *and a commitment value* $com \in \{0,1\}^{\lambda}$*.*

$\mathsf{open}(decom, com) \rightarrow \{\mathsf{msg}, \perp\}$**:** *The opening algorithm takes as input a decommitment value* $decom \in \{0,1\}^{\lambda}$ *and a commitment value* $com \in \{0,1\}^{\lambda}$ *and outputs either a message* msg *or* $\perp$ *in case com is not a valid commitment to any message.*

We require the following properties of a commitment scheme:

(Perfect) Completeness. We say that a commitment scheme is complete if for all messages $m \in \{0,1\}^{\lambda}$ *and* $(decom, com) \leftarrow \mathsf{com}(m)$*, we have that*

$$\Pr[\mathsf{open}(com, decom) = m] = 1.$$

Hiding. The hiding property means that no adversary can distinguish which of two messages are locked into the commitment. We say that $\mathcal{C}$ *is computationally/statistically* hiding *if,*

$$\Pr\left[b = b' \;\middle|\; \begin{array}{l} (\mathsf{msg}_0, \mathsf{msg}_1) \leftarrow \mathcal{A}(1^{\lambda}); \\ b \leftarrow \{0,1\}; \\ (decom, com) \leftarrow \mathsf{com}(\mathsf{msg}_b); \\ b' \leftarrow \mathcal{A}(com) \end{array} \right] \leq \frac{1}{2} + negl(\lambda)\ ,$$

is a negligible function for all ppt/unbounded adversaries $\mathcal{A}$*.*

Binding. Informally, binding says that the adversary cannot open the commitment in two different ways. We say that $\mathcal{C}$ is binding if no efficient adversary can find values $(com, decom_0, decom_1, \mathsf{msg}_0, \mathsf{msg}_1) \in \{0,1\}^\lambda$ with $\mathsf{msg}_0 \neq \mathsf{msg}_1$ such that $\texttt{open}(com, decom_0) = \mathsf{msg}_0$ and $\texttt{open}(com, decom_1) = \mathsf{msg}_1$.

Yao's Garbled Circuits Yao's garbled circuit [Yao82,BHR12] is a two-party secure computation scheme where the parties jointly compute a function output on private inputs. One party garbles the circuit, and the other party evaluates it.

Definition 13. *A projective garbling scheme consists of the following algorithms:*

$\texttt{garble}(1^\lambda, \texttt{C}) \to (\texttt{GC}, \mathbf{K})$**:** *The garbling algorithm* $\texttt{garble}$ *takes as input the security parameter λ and a boolean circuit $\texttt{C} : \{0,1\}^\ell \to \{0,1\}^m$, and it outputs a garbled circuit* $\texttt{GC}$ *and ℓ pairs of garbled labels $\mathbf{K} = (K_1^0, K_1^1, \ldots, K_\ell^0, K_\ell^1)$. For simplicity we assume that for every $i \in [\ell]$ and $b \in \{0,1\}$ it holds that $K_\ell^b \in \{0,1\}^\lambda$.*

$\texttt{eval}(\texttt{GC}, K_1, \ldots, K_\ell) \to y$**:** *The evaluation algorithm* $\texttt{eval}$ *takes as input the garbled circuit* $\texttt{GC}$ *and ℓ garbled labels $K_1, \ldots, K_\ell$, and outputs a value $y \in \{0,1\}^m$.*

We require the following properties of a projective garbling scheme:

Correctness. A garbling scheme $\varPi = (\texttt{garble}, \texttt{eval})$ is correct if for any boolean circuit $\texttt{C} : \{0,1\}^\ell \to \{0,1\}^m$ and $x = (x_1, \ldots, x_\ell)$ it holds that

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

where $(\texttt{GC}, \mathbf{K}) \leftarrow \texttt{garble}(1^\lambda, \texttt{C})$ with $\mathbf{K} = (K_1^0, K_1^1, \ldots, K_\ell^0, K_\ell^1)$, and $\mathbf{K}[x] = (K_1^{x_1}, \ldots, K_\ell^{x_\ell})$.

Next, we formally define the security notions we require for a garbling scheme.

Privacy. A garbling scheme is private if there exists a simulator $\texttt{simGC}$ *such that for every PPT adversary $\mathcal{A}$ it holds that*

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

where the experiment $\texttt{Expt}^{\texttt{priv}}_{\varPi,\mathcal{A},\texttt{simGC}}(\lambda, b)$ is defined as follows:

1. *The adversary $\mathcal{A}$ specifies $\texttt{C} : \{0,1\}^\ell \to \{0,1\}^m$ and $x = (x_1, \ldots, x_\ell) \in \{0,1\}^\ell$.*
2. *The challenger responds as follows:*
   – *If $b = 0$ set $(\texttt{GC}, \mathbf{K}) \leftarrow \texttt{garble}(1^\lambda, \texttt{C})$ with $\mathbf{K} = (K_1^0, K_1^1, \ldots, K_\ell^0, K_\ell^1)$. Responds with $(\texttt{GC}, \mathbf{K}[x])$ where $\mathbf{K}[x] = (K_1^{x_1}, \ldots, K_\ell^{x_\ell})$.*
   – *If $b = 1$ then respond with $(\texttt{GC}, K_1, \ldots, K_\ell) \leftarrow \texttt{simGC}(1^\lambda, \texttt{C}(x))$.*
3. *The adversary outputs a bit b' as the output of the experiment.*

Obliviousness. A garbling scheme is oblivious if there exists a simulator simGC *such that for every PPT adversary $\mathcal{A}$ it holds that*

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

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

1. *The adversary $\mathcal{A}$ specifies* $\mathtt{C}:\{0,1\}^\ell\to\{0,1\}^m$ *and* $x=(x_1,\ldots,x_\ell)\in\{0,1\}^\ell$.
2. *The challenger responds as follows:*
   – *If $b=0$ set* $(\mathtt{GC},\mathbf{K})\leftarrow\mathtt{garble}(1^\lambda,\mathtt{C})$ *with* $\mathbf{K}=(K_1^0,K_1^1,\ldots,K_\ell^0,K_\ell^1)$. *Responds with* $(\mathtt{GC},\mathbf{K})$.
   – *If $b=1$ then respond with* $(\mathtt{GC},K_1^0,K_1^1,\ldots,K_\ell^0,K_\ell^1)\leftarrow\mathtt{simGC}(1^\lambda)$.
3. *The adversary outputs a bit b' as the output of the experiment.*

Non-Interactive Key Exchange (NIKE) A non-interactive key-exchange protocol allows two parties to jointly compute a cryptographic key.

Definition 14. *A non-interactive key exchange scheme, parametrized by a distribution $\mathcal{D}$, consists of the following algorithms:*

$\mathtt{keygen}(1^\lambda)\to(\mathtt{pk},\mathtt{sk})$**:** *This algorithm produces a public - private key pair.*
$\mathtt{keyagree}(\mathtt{pk},\mathtt{sk})\to\mathtt{k}$**:** *This algorithm, given a public key and a secret key* from a different key pair*, produces a shared key for the holder of the public key and the holder of the secret key. Importantly, this algorithm should return the same key when run on P_i's public key and P_j's secret key, and vice versa.*

We require the following properties of a non-interactive key exchange scheme:

Correctness. We say that a non-interactive key exchange scheme is correct *if*

$$\Pr\begin{bmatrix}(\mathtt{pk}_0,\mathtt{sk}_0)\leftarrow\mathtt{keygen}(1^\lambda);\\(\mathtt{pk}_1,\mathtt{sk}_1)\leftarrow\mathtt{keygen}(1^\lambda);\\\mathtt{keyagree}(\mathtt{pk}_0,\mathtt{sk}_1)=\mathtt{keyagree}(\mathtt{pk}_1,\mathtt{sk}_0)\end{bmatrix}\geq 1-negl(\lambda)\ .$$

Security. We say that a non-interactive key exchange scheme is correct *if is* secure *if for all PPT adversaries $\mathcal{A}$,*

$$\Pr\left[b=b'\;\middle|\;\begin{array}{l}(\mathtt{pk}_0,\mathtt{sk}_0)\leftarrow\mathtt{keygen}(1^\lambda);\\(\mathtt{pk}_1,\mathtt{sk}_1)\leftarrow\mathtt{keygen}(1^\lambda);\\b\leftarrow\{0,1\};\\\mathtt{k}_0\leftarrow\mathtt{keyagree}(\mathtt{pk}_1,\mathtt{sk}_0);\\\mathtt{k}_1\leftarrow\mathcal{D};\\b'\leftarrow\mathcal{A}(\mathtt{pk}_0,\mathtt{pk}_1,\mathtt{k}_b)\end{array}\right]\leq\frac{1}{2}+negl(\lambda)\ .$$

Secret Sharing Scheme A t-out-of-n secret sharing scheme allows a party to "split" a secret into n shares that can be distributed among different parties. To reconstruct the original secret x at least t shares need to be supplied [CDN15].

Definition 15. *A t-out-of-n secret sharing scheme consists of the following algorithms:*

- $\texttt{share}(x) \to (s_1, \ldots, s_n)$*: The randomized algorithm* `share` *takes a secret x as input and output a sequence of shares.*
- $\texttt{reconstruct}(s_1, \ldots, s_\ell) \to x$*: The reconstruct algorithm* `reconstruct` *upon input a vector of ℓ shares outputs the secret x whenever $\ell \geq t$.*

We require the following properties of a t-out-of-n secret sharing scheme:

Privacy. The privacy property says that any combination of up to $t-1$ shares should leak no information about the secret x. Formally, for all (unbounded) adversaries $\mathcal{A}$, for any set $\mathcal{P}' = (P'_1, \ldots, P'_\ell) \subseteq \mathcal{P}$ such that $\ell < t$, and for every two secrets x_0, x_1,

$$\Pr\left[\mathcal{A}(s') = 1 : \left| \begin{matrix} s = \texttt{share}(x_0); \\ s' = (s_{P'_1}, \ldots, s_{P'_\ell}) \end{matrix} \right.\right] \equiv \Pr\left[\mathcal{A}(s') = 1 : \left| \begin{matrix} s = \texttt{share}(x_1); \\ s' = (s_{P'_1}, \ldots, s_{P'_\ell}) \end{matrix} \right.\right].$$

Shares simulatability. Additionally, we require the existence of an efficient simulator for the generated shares. Formally, there exists a PPT simulator `simshare` *and a negligible function $negl(\lambda)$ such that for every PPT adversary $\mathcal{A}$, every secret x and sufficiently large λ,*

$$|\Pr[\mathcal{A}(\texttt{share}(x)) = 1] - \Pr[\mathcal{A}(\texttt{simshare}(\lambda)) = 1]| \leq negl(\lambda).$$