Minimizing Setup in Broadcast-Optimal Two Round MPC

Ivan Damgård[1], Divya Ravi[1], Luisa Siniscalchi[1,2], and Sophia Yakoubov[1]*

[1] Aarhus University, Denmark; {ivan, divya, lsiniscalchi, sophia.yakoubov}@cs.au.dk
[2] Concordium Blockchain Research Center

Abstract. In this paper we consider two-round secure computation protocols which use different communication channels in different rounds: namely, protocols where broadcast is available in neither round, both rounds, only the first round, or only the second round. The prior works of Cohen, Garay and Zikas (Eurocrypt 2020) and Damgård, Magri, Ravi, Siniscalchi and Yakoubov (Crypto 2021) give tight characterizations of which security guarantees are achievable for various thresholds in each communication structure.
In this work, we introduce a new security notion, namely, *selective identifiable abort*, which guarantees that every honest party either obtains the output, or aborts identifying one corrupt party (where honest parties may potentially identify different corrupted parties). We investigate what broadcast patterns in two-round MPC allow achieving this guarantee across various settings (such as with or without PKI, with or without an honest majority).
Further, we determine what is possible in the honest majority setting *without* a PKI, closing a question left open by Damgård *et al.* We show that without a PKI, having an honest majority does not make it possible to achieve stronger security guarantees compared to the dishonest majority setting. However, if two-thirds of the parties are guaranteed to be honest, *identifiable abort* is additionally achievable using broadcast only in the second round.
We use fundamentally different techniques from the previous works to avoid relying on private communication in the first round when a PKI is not available, since assuming such private channels without the availability of public encryption keys is unrealistic. We also show that, somewhat surprisingly, the availability of private channels in the first round does not enable stronger security guarantees unless the corruption threshold is one.

Keywords: Secure Computation, Round Complexity, Minimal Setup

*Funded in part by the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 669255 (MPCPRO) and No 803096 (SPEC).

Table of Contents

1 Introduction

It is known that secure computation is possible in two rounds (whereas one round is clearly not enough). However, most known two-round protocols either only achieve the weakest security guarantee (selective abort) [ACGJ19], or achieve the best-possible guarantee in their setting by making use of a broadcast channel in both rounds [GLS15,GS18,BL18]. Implementing broadcast via a protocol among the parties makes no sense in this setting, as the resulting protocol would require much more than two rounds. However, broadcast can also be done using physical assumptions or external services such as blockchains. This typically means that broadcast is expensive and/or slow, so it is important to try to minimize the usage of broadcast (while achieving as strong a security guarantee as possible).

Before discussing previous work in this direction and our contribution, we establish some useful termonology.

1.1 Terminology

In this work, we categorize protocols in terms of (a) the kinds of communication required in each round, (b) the security guarantees they achieve, (c) the setup they require, and (d) the corruption threshold t they support. We will use shorthand for all of these classifications to make our discussions less cumbersome.

Communication Structure We refer to protocols that use two rounds of broadcast as BC-BC; protocols that use broadcast in the first round only as BC-P2P; protocols that use broadcast in the second round only as P2P-BC; and protocols that don't use broadcast at all as P2P-P2P.

Note that, when no PKI is available, it is not realistic to assume *private* channels in the first round since it is unclear how such private channels would be realized in practice without public keys. Therefore, in what follows, "P2P" in the first round refers to *open* peer-to-peer channels which an adversary can listen in on [3] – unless we explicitly state otherwise. We do assume the availability of private channels in the second round, since one can broadcast (or send over peer-to-peer channels) an encryption under a public key received in the first round.

Security Guarantees There are six notions of security that a secure computation protocol could hope to achieve, described informally below.

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

Selective Identifiable Abort (SIA): A secure computation protocol achieves *selective identifiable abort* if every honest party either obtains the output, or aborts identifying one corrupt party (where the corrupt party identified by different honest parties may potentially be different).

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

All the notions above, except SIA have been studied previously. This new notion that we introduce here is strictly stronger than selective abort but weaker than identifiable abort (which demands that all honest parties must be in agreement). In the light of the results we prove, we believe that this is an interesting conceptual contribution, as it gives the possibility to the honest player to identify (at least partially) who prevented everybody from retrieving the output.

Selective abort is the weakest notion of security, and is implied by all the others; guaranteed output delivery is the strongest, and implies all the others. Notably, fairness and identifiable abort are incomparable; selective identifiable abort and unanimous abort are incomparable as well.

Setup The following forms of setup, from strongest to weakest, are commonly considered in the MPC literature:

[3] We do assume that the peer-to-peer channels are authenticated.

Correlated randomness (CR), where the parties are given input-independent secrets which may be correlated,

A public key infrastructure (PKI), where each party has an independent honestly generated [4] public-secret key pair where the public key is known to everyone, and

A common reference string (CRS), where no per-party information is available, but a single trusted reference string is given.

We focus primarily on protocols that only use a CRS, which is the weakest form of setup (except for the extreme case of no setup at all). To make our prose more readable, when talking about e.g. a secure computation protocol that achieves security with identifiable abort given a CRS and uses broadcast in the second round only, we will refer to it as a P2P-BC, IA, CRS protocol. If we additionally want to specify the corruption threshold t to be x, we call it a P2P-BC, IA, CRS, $t \leq x$ protocol.

1.2 Prior Work

Cohen, Garay and Zikas [CGZ20] initiated the study of two-round secure computation with broadcast available in one, but not both, rounds. They showed that, in the P2P-BC setting, UA is possible even given a dishonest majority, and that it is the strongest achievable guarantee in this setting. They also showed that, in the BC-P2P setting, SA is the strongest achievable security guarantee given a dishonest majority.

The subsequent work by Damgård, Magri, Ravi, Siniscalchi and Yakoubov [DMR+21] continued this line of inquiry, focusing on the honest majority setting. They showed that given an honest majority, in the P2P-BC setting IA is achievable (but fairness is not), and in the BC-P2P setting, the strongest security guarantee — GOD — is achievable.

The constructions of Cohen *et al.* do not explicitly use a PKI, but they do rely on private communication in the first round, which in practice requires a PKI, as discussed above. The constructions of Damgård *et al.* rely on a PKI even more heavily. The natural open question therefore is: what can be done assuming no PKI — only a CRS, and no private communication in the first round?

We note that the recent work of Goel, Jain, Prabhakaran and Raghunath [GJPR21] considers instead the plain model or the availability only of a bare PKI (where it is assumed that corrupt parties may generate their public key maliciously). They show that in plain model, in the absence of private channels, no secure computation is possible even given an honest majority. Further, given broadcast (in both rounds) IA is impossible in the plain model, while the strongest guarantee of GOD is feasible in the bare PKI model. Our model is incomparable to that of Goel *et al.* since we consider the availability of a CRS, and communication patterns where broadcast is limited to one of the two rounds.

[4] Throughout this paper, we use the term 'PKI' to refer to a 'trusted PKI', where the PKI keys are assumed to be honestly generated for all parties.

To the best of our knowledge, the notion of selective identifiable abort is not discussed in previous work.

1.3 Our Contributions

We summarize the contributions of our work in two broad categories, described below.

1.3.1 Introduction of SIA

We introduce and formalize a new security notion of MPC protocols, that we refer to as selective identifiable abort (SIA). Further, we investigate the feasibility of two-round SIA MPC protocols with different broadcast patterns for various settings – with or without PKI, and with or without honest majority. As it turns out, SIA is an interesting notion because it can be achieved in cases where previously only weaker or incomparable notions were known to be possible. Notably, for the P2P-P2P or BC-P2P and $t < n/3$ settings, SIA can be achieved and is the best possible guarantee, where previously only selective abort was known. Note that, with only selective abort, stopping honest players from getting the output is basically without consequences for the adversary, while with SIA, each honest player will identify at least one corrupt player.

In the following we explain all the results on SIA in more detail: In Theorem 3, we show that any BC-BC (respectively P2P-BC) protocol (with some additional properties) can be transformed to an SIA protocol for the same corruption threshold where the second round communication is over peer-to-peer channels. Plugging in the appropriate IA protocols to this theorem yields several positive results, summarized in Table 1 (for the CRS setting) and Section 1.4.3 (for the PKI setting). Namely, we obtain that when we assume only a CRS and no private communication in the first round, SIA is achievable in the P2P-P2P setting with $t < \frac{n}{3}$ and in the BC-P2P setting with $t < n$; finally when a PKI is available, SIA is also possible in the P2P-P2P setting with $t < \frac{n}{2}$.

In light of the above, what remains to be investigated are patterns where broadcast is not available in the first-round, for settings with only a CRS and honest majority ($\frac{n}{2} > t \geq \frac{n}{3}$); and settings with PKI and dishonest majority ($\frac{n}{2} \leq t < n$).

In the CRS only setting, we show that P2P-BC, SIA is impossible to achieve even with an honest majority (Theorem 4, this result holds even when the first-round communication is private). Finally, we observe that the impossibility of P2P-BC, IA, PKI protocols for $t < n$ in [CGZ20] can be extended to SIA as well (see Obs 1).

1.3.2 Complete Characterization of two-round MPC in the CRS model with honest majority

Assuming only a CRS and no private communication in the first round, we give a complete characterization of what can be done in two rounds with respect to all the other security guarantees and different broadcast patterns.

In a nutshell, we show that assuming only a CRS, an honest majority does not give much of an advantage over a dishonest majority: regardless of the corruption threshold, IA continues to remain impossible in the P2P-BC setting (directly follows from impossibility of SIA in this setting, Theorem 4) and in the BC-P2P setting, UA continues to remain impossible (Theorem 5). [5]. The latter extends the impossibility result of Patra and Ravi [PR18], which holds for $n \leq 3t$ (but does not hold for $t > 1$ and any n).

However, if at least two thirds of the parties are honest, in the P2P-BC setting IA is additionally possible (Theorem 2). To show this we give a construction based on a new primitive called *one-or-nothing secret sharing with intermediaries* (adapted from one-or-nothing secret sharing [DMR+21]), which may be of independent interest.

Most of our lower bounds hold even given private communication in the first round; however, our constructions do not require it. This shows that surprisingly, in most cases, having private communication in the first round cannot help achieve stronger guarantees.

The one exception is the case where the adversary can only corrupt one party (that is, $t = 1$); for $t = 1$ and $n \geq 4$, guaranteed output delivery can be achieved given private channels in the first round [IKP10,IKKP15] even when broadcast is completely unavailable. However, we show that without private channels in the first round fairness (and thus also guaranteed output delivery) is unachievable (Cor 2), even if broadcast is available in both rounds and the adversary corrupts only one participant [6]. We also show that without private channels in the first round, if broadcast is unavailable in the second round, unanimous abort is unachievable (Cor 1).

Finally, we make a relatively simple observation, showing that the positive results from Cohen *et al.* still hold, even without private communication in the first round.

We summarize our findings in Table 1 , and the special case of $t = 1$ in Table 2.

1.4 Technical Overview

In Section 1.4.1, we summarize our lower bounds; in Section 1.4.2, we summarize our constructions. These results assume a setup with CRS only. Lastly, in Section 1.4.3, we summarize the results related to SIA, when PKI is available.

1.4.1 Lower Bounds We present several lower bounds, some of which hold even when private channels are available in the first round. This is in contrast

[5] Given an additional round of communication instead of a PKI, things look different; Badrinarayanan *et al.* [BMMR21] study broadcast-optimal *three*-round MPC with GOD given an honest majority and CRS, and show that GOD is achievable in the BC-BC-P2P setting.

[6] This strengthens the fairness impossibility result of Gordon *et al.* [GLS15] which holds for $n \leq 3t$.

Without PKI, in Two Rounds

Broadcast Pattern R1	Broadcast Pattern R2	t	Selective Abort (SA)	Selective Identifiable Abort (SIA)	Unanimous Abort (UA)	Identifiable Abort (IA)	Fairness (FAIR)	Guaranteed Output Delivery (GOD)
BC	BC	$\frac{n}{2} \leq t < n$	✓	✓	✓	✓[CGZ20] w.m.c	✗[Cle86]	✗
P2P	BC		✓	✗[CGZ20] (Obs 1)	✓[CGZ20] w.m.c	✗	✗	✗
BC	P2P		✓	✓[CGZ20] w.m.c, last round P2P (Theorem 3)	✗[CGZ20]	✗	✗	✗
P2P	P2P		✓[CGZ20] w.m.c	✗	✗	✗	✗	✗
BC	BC	$\frac{n}{3} \leq t < \frac{n}{2}$	✓	✓	✓	✓[CGZ20] w.m.c	✗[PR18,GLS15]	✗
P2P	BC		✓	✗Theorem 4	✓[CGZ20] w.m.c	✗	✗	✗
BC	P2P		✓	✓[CGZ20] w.m.c, last round P2P (Theorem 3)	✗[PR18]	✗	✗[PR18,GLS15]	✗
P2P	P2P		✓[CGZ20] w.m.c	✗	✗	✗[CL14]	✗	✗[LSP82]
BC	BC	$t < \frac{n}{3}$	✓	✓	✓	✓[CGZ20] w.m.c	✗ for $t > 1$ [GIKR02]	✗ for $t > 1$
P2P	BC		✓	✓	✓[CGZ20] w.m.c	✓ Theorem 2	✗ for $t > 1$	✗ for $t > 1$
BC	P2P		✓	✓[CGZ20] w.m.c, last round P2P (Theorem 3)	✗ for $t > 1$ (Theorem 5)	✗ for $t > 1$	✗ for $t > 1$ [GIKR02]	✗ for $t > 1$
P2P	P2P		✓[CGZ20] w.m.c	✓ Theorem 2, last round P2P (Theorem 3)	✗ for $t > 1$ [DMR+21]	✗ for $t > 1$	✗ for $t > 1$	✗ for $t > 1$

Table 1: Feasibility and impossibility for two-round MPC with different guarantees and broadcast patterns when only a CRS is available (but no PKI or correlated randomness). The R1 column describes whether broadcast is available in round 1; the R2 column describes whether broadcast is available in round 2. In our constructions, round 1 communications are not private; negative results hold even with private round 1 communications. 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. Beige table cells are lower bounds; green table cells are upper bounds.

The $t = 1$ Case

Broadcast Pattern R1	Broadcast Pattern R2	t	Selective Abort (SA)	Selective Identifiable Abort (SIA)	Unanimous Abort (UA)	Identifiable Abort (IA)	Fairness (FAIR)	Guaranteed Output Delivery (GOD)
Without Private Channels in Round 1:								
BC	BC	$t = 1,\ n > 1$	Table 1	Table 1	Table 1	Table 1	✗Cor 2 → ↓	✗
P2P	BC		Table 1	Table 1	Table 1	Table 1	✗ →	✗
BC	P2P		Table 1	Table 1	✗Cor 1 → ↓	✗	✗Cor 1, 2 → ↓	✗
P2P	P2P		Table 1	Table 1	✗ →	✗	✗ →	✗
With Private Channels in Round 1:								
Any		$t = 1,\ n = 4$	✓ ←	✓	✓ ←	✓[IKKP15]	✓ ←	✓[IKKP15]
Any		$t = 1,\ n \geq 5$	✓ ←	✓	✓ ←	✓[IKP10]	✓ ←	✓[IKP10]

Table 2: Feasibility and impossibility for two-round MPC with different guarantees and broadcast patterns when only a CRS is available, when $t = 1$. We refer to Table 1 for the cases already covered therein.

to our constructions which avoid the use of private channels before the parties had a chance to exchange public keys.

With private channels In Section 4.1, we present two main lower bounds that hold even if private channels are available in the first round.

Our first lower bound (Theorem 4) shows that P2P-BC, SIA, CRS protocol is impossible when $n \leq 3t$. To show this, we consider a hypothetical 3-party P2P-BC, SIA, CRS protocol where an adversary who controls just one party, say P behaves inconsistently over the first-round peer-to-peer channels and then chooses to act in the second round based on the information sent to one of the honest parties, say P'. Then, SIA guarantees that P' must compute the output even though she finds the pair of remaining parties in conflict, as she cannot decide whom to blame. This makes the protocol vulnerable to an attack by potentially corrupt P' who can simulate this kind of conflict in her head by recomputing the messages of P based on inputs of her choice. Infact, this argument can be extended for $n \leq 3t$.

Our second lower bound (Theorem 5) shows that BC-P2P, UA, CRS protocol is impossible when $t > 1$. To show this, we argue that in any hypothetical BC-P2P, UA, CRS protocol, an adversary who is able to control just two parties is able to perform an even more powerful attack: after execution, she is able to recompute the function output locally on corrupt party inputs of her choice (together with the same fixed set of honest party inputs). This is called a *residual function attack*. This completes the overview of the lower bounds that hold when private channels are present.

When $t = 1$, we show that the availability of private channels makes a difference. When private channels are available in the first round, the strongest guarantee — guaranteed output delivery — is known to be achievable as long as $n \geq 4$ [IKP10,IKKP15]. However, we show in Section 4.2 and outline below that without private channels in the first round, the landscape is quite different.

Without private channels In this setting, an adversary can observe all messages sent by an honest party P in the first round; so, those first-round messages cannot suffice to compute the function on P's input — P's second-round messages are crucially necessary for this. If P's first-round messages were enough, the adversary would be able to mount a residual function attack: given P's first-round messages, the adversary would be able to compute the function on P's input (along with inputs of her choice on behalf of the other parties) in her head, by simulating all the other parties. However, if we aim for either unanimous abort (without use of broadcast in the second round) or fairness, we can also argue that P's second-round messages *cannot* be necessary. If we would like to achieve unanimous abort without use of broadcast in the second round, it is important that the adversary not be able to break unanimity by sending different second-round messages to different parties. If we would like to achieve fairness, it is similarly important that the adversary not be able to deny the honest parties access to the output by withholding her second-round messages. So, to achieve either of those goals, the second-round message both must and cannot matter; we thus rule out BC-P2P, UA, CRS protocols (Cor 1) and BC-BC, FAIR, CRS protocols (Cor 2) when no private channels are available in the first round.

1.4.2 Upper Bounds

Feasibility of P2P-BC, IA, CRS when $t < \frac{n}{3}$ In Section 3.2, we present our main positive result, which is a P2P-BC, IA, CRS, $t < \frac{n}{3}$ construction (Figure 3.2). Our construction builds on the construction of Damgård *et al.* [DMR+21] (which, in turn, builds on the construction of Cohen *et al.* [CGZ20]). Like those prior works, we take a protocol that requires two rounds of broadcast, and compile it. Since broadcast is only available in the second round, the key is to ensure that a corrupt party can't break the security of the underlying protocol by sending inconsistent messages to different honest parties in the first round. The solution is to delay computation of the second round messages until parties are sure they agree on what was said in the first round.

Following previous work, we do this by having each party G garble her second-message function (which takes as input all the first-round messages that party expects to receive) and broadcast that garbled circuit in the second round. G additionally secret shares all of the labels for her garbled circuit. We can get identifiable abort from this if we make sure that one of two things happen: (a) sufficiently many parties receive a given first-round message bit coming from a sender S, implying that the label corresponding to that bit is reconstructed (unanimously, over broadcast); or (b) someone is unanimously identified as a cheater. (Of course, two labels for the same input wire should never be reconstructed, since this would compromise the security of the garbled circuit scheme.)

To achieve this, Damgård *et al.* introduce (and use in their construction) the notion of *one-or-nothing secret sharing*. Unfortunately, this primitive crucially relies on a PKI: in the second round, each player must be able to prove that she received a certain message from S in the first round (or abstain if she received nothing). Given a PKI, this can be done by having S sign her first-round messages. Of course, without a PKI, this cannot work as there is no time to agree on public keys.

Therefore, without a PKI, we need a different approach. The approach we use is instead to check in the second round whether there is sufficient consensus among the parties about what S sent in the first round, and only reconstruct the corresponding labels if this is the case. To this end, we define a new primitive in the CRS model called *one-or-nothing secret sharing with intermediaries*. In such a scheme, each garbler G performs two layers of Shamir sharing: first, each label is shared, creating for each party R a share s_{R}. Second, each s_{R} is shared among all parties. Everyone now acts as intermediaries, and passes their sub-shares of s_{R} on to R in the second round. This ensures that a corrupt G cannot fail to deliver a share to R, since G cannot fail to communicate with more than t intermediaries without being identified. Simultaneously, each participant R broadcasts a message enabling the public recovery of only the label share corresponding to what she received from S in the first round. Enough shares for a given label are only recoverable if enough participants received the same bit from S, implicitly implementing the consensus check we mentioned above.

There is one final caveat we need to take care of: the standard network model assumes peer-to-peer "open" channels where the adversary can observe all messages sent. With a PKI, we can make use of private channels (even in the first round), by using public-key encryption (PKE). However, in the absence of a PKI, this makes little sense, so we should *not* use private channels in the first round. Under this constraint we cannot send shares of secrets in the first round. So, we need to figure out a way for G to send sub-shares of R's share s_{R} to intermediaries, and for intermediaries to pass these sub-shares on to R, in a *single* round of broadcast.

The approach we use is as follows: in the second round, for each sub-share of s_{R} intended for intermediary I, G will broadcast an encryption c of that sub-share, under a public key received from I in the first round. Simultaneously, I passes on all of sub-shares to R by broadcasting *transfer keys*. Depending on which value should be decrypted, R broadcasts the relevant decryption key which enables the recovery of the corresponding plaintext. We informally refer to this approach as *transferrable encryption*, where a party is able to transfer decryption capabilities to another, even without first seeing the ciphertext in question.

Our construction of *one-or-nothing secret sharing with intermediaries* relies on a CPA-secure PKE scheme and non-interactive zero-knowledge (NIZK) proof system. This is used as a building block in our P2P-BC, IA, CRS, $3t < n$ construction (formalized in Figure 3.2) following the above blueprint.

Modifying Prior P2P-BC Constructions The work of Cohen *et al.* gives a construction in the P2P-BC and P2P-P2P settings that uses only a CRS (not a PKI); however, they use private communication in the first round. We observe that we can modify their construction in a straightforward way to only use *public* peer-to-peer communication in the first round, which is more realistic without a PKI. Their construction is a compiler, and in the first round, two things are sent: messages from the underlying construction; and (full-threshold) secret shares of garbled circuit labels, which need to be communicated privately, and which are then selectively published in the second round. Let's pick an underlying construction that uses public communication only (e.g. the construction of [GS18]). Now, to avoid private communication in the first round, we modify the protocol to delay secret sharing until the second round. Instead, the only additional thing the parties do in the first round is exchange public encryption keys. Like in our construction (described above), it might look like delaying secret sharing poses a problem, since the share recipients need to broadcast the relevant shares to enable output recovery, but if they only receive their shares in the second (last) round, they don't have time to do this. So, we have the share sender G encrypt each share meant for receiver R under a one-time public key belonging to R. Simultaneously, R will publish the corresponding secret key if and only if she wishes to enable the reconstruction of that label.[7] In this way, the same guarantees can be achieved without using private communication in the first round.

Feasibility Results for SIA In Section 3.3, we argue that a BC-BC protocol (respectively a P2P-BC) Π_{bc} that securely computes f with identifiable abort can be turned into an SIA protocol Π (with the same corruption threshold) where the second round is run over peer-to-peer channels, as long as Π_{bc} satisfies the following two properties: 1) the simulator can extract inputs from the first-round; 2) it is efficient to check whether a given second-round of the protocol is correct.

The protocol Π works in the same way as Π_{bc}, except that the second round is sent over peer-to-peer channels. Intuitively, the only advantage that the adversary has in Π is to send inconsistent last round messages. However, we argue that this cannot lead to a pair of honest parties obtaining two different non-$\bot$ outputs. This is because of our assumption that the simulator of Π_{bc} extracts input from the first round messages (and say receives the output y from the ideal functionality). This means that no matter what second round messages the adversary sends in Π_{bc}, the output can never be $y' \neq \bot$ such that $y' \neq y$. More specifically, the adversary's second round messages in Π_{bc} can only determine whether all the honest parties learn y or all the honest parties learn the identity of a cheater (can be potentially chosen by the adversary during the

[7] Note that the full power of our one-or-nothing secret sharing with intermediaries is not necessary here; in our construction, we only require two levels of sharing and intermediaries in order to achieve *identifiable* abort, while this construction aims only for selective and unanimous abort in the two different settings respectively.

second broadcast round in $\Pi_{\mathtt{bc}}$, say, by making a corrupt party stop sending messages or sending invalid messages in the second round). Since these second round messages are now sent over peer-to-peer channels instead (but it is possible to efficiently check their validity), we can conclude that each honest party in Π would either learn the output y or the identity of a cheater (depending on the version of the second round message the adversary sends privately). It may be the case that honest parties learn different cheaters or some of them learn the output y while others don't; however, this suffices for SIA guarantee.

In Section 3.3 we give candidate constructions of BC-BC protocol (respectively a P2P-BC) with identifiable abort, that have the additional properties described above and can thereby be used to yield the BC-P2P (respectively P2P-P2P) SIA upper bounds.

1.4.3 Completing the picture of SIA with PKI

Given that the notion of selective identifiable abort is introduced in this work, we also investigate how it affects the landscape when a PKI is available. This setting was studied by Damgård *et al.* [DMR+21] for the case of honest majority and Cohen *et al.* [CGZ20] for the case of dishonest majority.

The case of BC-BC is already settled by Cohen *et al.*, who give an IA construction (stronger than SIA) for $t < n$, relying just on CRS. Next, we note that our observation in Section 3.3 lets us transform the above into an SIA protocol (with the same corruption threshold) where the second round is run over peer-to-peer channels; settling the case of BC-P2P setting.

In the P2P-BC setting, we observe that the impossibility of P2P-BC, IA, PKI protocols for $t < n$ in Cohen *et al.* can be extended to SIA as well (outlined in Section F). However, assuming an honest majority ($t < \frac{n}{2}$), feasibility of SIA follows directly from the P2P-BC, IA, PKI construction of Damgård *et al.*. Applying the observation in Section 3.3 to this IA construction of Damgård *et al.*, let us achieve SIA for P2P-P2P setting with the same threshold.

This settles the question of feasibility of two-round SIA with various broadcast patterns in the PKI setting.

1.5 Broadcast Complexity

In the previous two works, no attempt was made to minimize the broadcast overhead of the compilers. They all require the broadcast of garbled second-message functions, the size of which often scales with the complexity of the function computed, which is potentially large. We observe that a generic broadcast optimization (which is folklore, and has appeared in some previous work [HR14,GP16,CCD+20]) can be applied to any message which is already known to the sender in the first round, but need not be broadcast until the second round. Using this optimization, the size of the additional broadcasts that our compiler — and the compilers of Cohen *et al.* and Damgård *et al.*— becomes independent of the size of the function being computed.

The broadcast optimization is quite straightforward. It enables reliable broadcast of arbitrarily long messages, while only sending fixed-length messages over the broadcast channel in the second round. The dealer sends its message to all the recipients over peer-to-peer channels in the first round. Each recipient then echos the message it received *over peer-to-peer channels* in the second round. Finally, in the second round, each party also broadcasts a *hash* of the message. If there exists a majority of parties who broadcast the same hash h, then each honest party outputs a pre-image of h. (Each party must have received a pre-image of h because at least one of the broadcasters of h must be honest.) Otherwise, honest parties blame the dealer. Only hashes are sent over the broadcast channel, and the size of those hashes is independent of the size of the message.

Finally, we note that when applying this optimization to our construction, and that of Cohen *et al.* and Damgård *et al.*, garbled circuits which were previously not broadcast until the second round are now sent (over peer-to-peer channels) in the first round. This necessitates the use of adaptive garbled circuits[8].

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.[9] We also give the formal definition of the new security notion of selective identifiable abort (sl-idabort).

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

The protocol is executed in rounds (i.e., the protocol is synchronous). Each round consists of the send phase and the receive phase, where parties can respectively send the messages from this round to other parties and receive messages from other parties. In every round parties can communicate either over a broadcast channel or a fully connected peer-to-peer (P2P) network. If peer-to-peer communication occurs in the first round without a PKI, we assume these channels are "open"; that is, the adversary sees all messages sent.[10] In other cases, we assume that these channels can be private, since communications can be encrypted using public keys that are either available via a PKI or exchanged in the first round. In all cases, we assume the channels to be ideally authenticated.

[8] Adaptive garbling schemes [BHR12a] remain secure against an adversary who obtains the garbled circuit and then selects the input.

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

[10] Some of our negative results hold even if private peer-to-peer channels are available in the first round. However, our positive results do not make use of such channels.

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

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

Definition 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 (sl-abort), selective identifiable abort (sl-idabort), unanimous abort (un-abort), identifiable abort (id-abort), fairness (fairness) and guaranteed output delivery (god).

Definition 2 (Ideal Computation). *Consider* $\mathsf{type} \in \{\mathsf{sl\text{-}abort}, \mathsf{un\text{-}abort}, \mathsf{sl\text{-}idabort}, \mathsf{id\text{-}abort}, \mathsf{fairness}, \mathsf{god}\}$. *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 corrupt parties. Then, the joint ideal execution of f under $(\mathcal{S}, \mathcal{I})$ on input vector $x = (x_1, \ldots, x_n)$, auxiliary input* aux *to $\mathcal{S}$ and 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 simulator $\mathcal{S}$ may send to the trusted party arbitrary inputs for the corrupt parties. Let x'_i be the value actually sent as the input of party P_i.*
2. Trusted party speaks to simulator*: The trusted party computes $(y_1, \ldots, y_n) = f(x'_1, \ldots, x'_n)$. If there are no corrupt parties or $\mathsf{type} = \mathsf{god}$, proceed to step 4.*
   (a) If $\mathsf{type} \in \{\mathsf{sl\text{-}abort}, \mathsf{un\text{-}abort}, \mathsf{sl\text{-}idabort}, \mathsf{id\text{-}abort}\}$: *The trusted party sends $\{y_i\}_{i\in\mathcal{I}}$ to $\mathcal{S}$.*
   (b) If $\mathsf{type} = \mathsf{fairness}$: *The trusted party sends* ready *to $\mathcal{S}$.*
3. Simulator $\mathcal{S}$ responds to trusted party*:*
   (a) If $\mathsf{type} = \mathsf{sl\text{-}abort}$: *The simulator $\mathcal{S}$ can select a set of parties that will not get the output as $\mathcal{J} \subseteq [n] \setminus \mathcal{I}$. (Note that $\mathcal{J}$ can be empty, allowing all parties to obtain the output.) It sends $(\mathtt{abort}, \mathcal{J})$ to the trusted party.*
   (b) If $\mathsf{type} \in \{\mathsf{un\text{-}abort}, \mathsf{fairness}\}$: *The simulator can send* `abort` *to the trusted party. If it does, we take $\mathcal{J} = [n] \setminus \mathcal{I}$.*
   (c) If $\mathsf{type} = \mathsf{sl\text{-}idabort}$: *The simulator $\mathcal{S}$ can select a set of parties that will not get the output as $\mathcal{J} \subseteq [n] \setminus \mathcal{I}$. (Note that $\mathcal{J}$ can be empty, allowing all parties to obtain the output.) For each party j in $\mathcal{J}$, the adversary*

selects a corrupt party $i_j^* \in \mathcal{I}$ who will be blamed by party j. It sends $(\texttt{abort}, \mathcal{J}, \{j, i_j^*\}_{j\in\mathcal{J}})$ to the trusted party.

(d) If $\mathsf{type} = \mathsf{id\text{-}abort}$: *If it chooses to abort, the simulator $\mathcal{S}$ can select a corrupt party $i^* \in \mathcal{I}$ who will be blamed, and send $(\texttt{abort}, i^*)$ 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* abort *from the simulator $\mathcal{S}$,*
      i. *It sets the abort message* abortmsg*, as follows:*
         – *if* $\mathsf{type} \in \{\mathsf{sl\text{-}abort}, \mathsf{un\text{-}abort}, \mathsf{fairness}\}$*, we let* $\mathsf{abortmsg} = \bot$*.*
         – *if* $\mathsf{type} = \mathsf{sl\text{-}idabort}$*, we let* $\mathsf{abortmsg} = \{\mathsf{abortmsg}_j\}_{j\in\mathcal{J}} = (\bot, i_j^*)_{j\in\mathcal{J}}$*.*
         – *if* $\mathsf{type} = \mathsf{id\text{-}abort}$*, we let* $\mathsf{abortmsg} = (\bot, i^*)$*.*
      ii. *The trusted party sends y_j to every party P_j, $j \in [n] \setminus \mathcal{J}$.*
      If $\mathsf{type} = \mathsf{sl\text{-}idabort}$*, the trusted party then sends* $\mathsf{abortmsg}_j$ *to each party P_j, $j \in \mathcal{J}$; otherwise, the trusted party sends* abortmsg *to every party P_j, $j \in \mathcal{J}$*

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

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

Definition 3. *Consider* $\mathsf{type} \in \{\mathsf{sl\text{-}abort}, \mathsf{un\text{-}abort}, \mathsf{sl\text{-}idabort}, \mathsf{id\text{-}abort}, \mathsf{fairness}, \mathsf{god}\}$*. 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 $\mathcal{A}$ with auxiliary input* aux*, there exists a PPT simulator $\mathcal{S}$ such that for every $\mathcal{I} \subseteq [n]$ of size at most t, for all $x \in (\{0,1\}^*)^n$, for all large enough $\lambda \in \mathbb{N}$, it holds that*

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

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: $\texttt{frst-msg}_i$, to compute P_i's first messages to its peers; $\texttt{snd-msg}_i$, to compute P_i's second messages; and $\texttt{output}_i$, to compute P_i's output. Thus, a protocol Π can be defined as $\{(\texttt{frst-msg}_i, \texttt{snd-msg}_i, \texttt{output}_i)\}_{i\in[n]}$.

The syntax of the algorithms is as follows:

– $\texttt{frst-msg}_i(x_i, r_i) \to (\mathsf{msg}^1_{i\to 1}, \ldots, \mathsf{msg}^1_{i\to n})$ 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.

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

We implicitly assume that all of these algorithms also take a CRS as input when one is available.

When the first round is over broadcast channels, we consider $\texttt{frst-msg}_i$ to return only one message — msg^1_i. Similarly, when the second round is over broadcast channels, we consider $\texttt{snd-msg}_i$ to return only msg^2_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 Upper Bounds

We begin with a description of our new primitive, one-or-nothing secret sharing with intermediaries, which is used as a building block in our IA construction. Next, we present our positive results for IA and SIA.

3.1 One-or-Nothing Secret Sharing with Intermediaries

Damgård *et al.* [DMR+21] introduce one-or-nothing secret sharing, which allows a dealer to share a vector of secrets in such a way that during reconstruction, at most one of the secrets is recovered (the share holders essentially vote on which one). The correctness guarantee is that if sufficiently many share holders vote for a certain index, and no-one votes against that index (though some parties may equivocate), the value at that index is recovered; the security guarantee is that if at least one party votes for a certain index, the adversary learns nothing about the values at any *other* index. Damgård *et al.* present two versions of this primitive: the default version, and a *non-interactive* version, where parties can vote even if they have not received a share from the dealer. This is done by assuming the dealer shares secret keys with each party, which can be realized via non-interactive key exchange, using a PKI.

Unfortunately, this non-interactive one-or-nothing secret sharing tool (referred to as `1or0`) does not extend to a setting where no PKI is available. In the absence of PKI, the main challenge is to ensure that the share intended for a party, say P, gets delivered (so that her share corresponding to the secret at the index she votes for can be recovered). We achieve this by modeling the fact that other parties can be intermediaries who aid this share transfer. For the setting where only a CRS is available, we propose a new variant of one-or-nothing secret sharing: namely, one-or-nothing secret sharing with intermediaries (referred to as `1or0wi`).

In order to simplify the presentation of our P2P-BC, IA, CRS construction, we define one-or-nothing secret sharing with intermediaries as a *maliciously-secure* primitive. The first round of our protocol is reserved for the exchange of public keys, so sharing and reconstruction must take place in a single round. The definitions of one-or-nothing secret sharing with intermediaries capture the fact that keys may not have been exchanged consistently, but demand that reconstruction succeeds if blame cannot be assigned. We discuss the syntax, definitions and construction of one-or-nothing secret sharing with intermediaries below.

3.1.1 Syntax A one-or-nothing secret sharing scheme [DMR+21] consists of four algorithms: `setup`, `share`, `vote`, and `reconstruct`. `setup` returns a shared secret key belonging to the dealer and one of the receivers; these keys are then used within `share`, and again in `vote`. To make our one-or-nothing secret sharing with intermediaries secure against malicious adversaries, we move to a public-key syntax, which makes it easier to check parties' behavior using zero knowledge proofs. We change `setup` to return a common reference string crs; keys are then produced by `keygen`, which creates a key pair for one of the receivers. `share`, `vote` and `reconstruct` now all expect the receivers' public keys as input. The syntax of `reconstruct` is modified to support cheater identification; if sufficiently many (at least $n - t$) parties vote for the same value, then either the secret corresponding to this value will be reconstructed, or a cheating party will be identified. We present the syntax of the maliciously-secure one-or-nothing secret sharing with intermediaries below.

$\texttt{setup}(1^\lambda) \to crs$ is an algorithm which takes as input the security parameter and generates the common reference string.
$\texttt{keygen}(crs) \to (\texttt{sk}, \texttt{pk})$ is an algorithm which takes as input the common reference string and generates a key pair.
$\texttt{share}(crs, \texttt{pk}_1, \ldots, \texttt{pk}_n, \texttt{z}^{(1)}, \ldots, \texttt{z}^{(l)}) \to s$ is an algorithm run by the dealer D which takes as input all the parties' public keys, and the l values that are being shared. It outputs a single share s.
$\texttt{vote}(crs, \texttt{sk}_i, \texttt{pk}_1, \ldots, \texttt{pk}_n, v_i) \to \overline{s}_i$ is an algorithm run by party i which takes as input party i's secret key, all the parties' public keys, and a vote v_i, where $v_i \in \{1, \ldots, l, \perp\}$ can either be an index of a value, or it can be $\perp$ if party i is unsure which value it wants to vote for. It returns a ballot $\overline{s}_i$.
Note that, to allow `share` and `vote` to be executed in a single round, `vote` does not take as input the share s.
$\texttt{reconstruct}(crs, s, (\texttt{pk}_1, v_1, \overline{s}_1), \ldots, (\texttt{pk}_n, v_n, \overline{s}_n)) \to \{\texttt{z}^{(v)}, \perp, \perp_i\}$ is an algorithm which takes as input the output of `share` run by the dealer D, the outputs of `vote` run by each of the n parties, as well as their votes, and outputs the value $\texttt{z}^{(v)}$ which received a majority of votes, *or* $\perp$, *or* $\perp_i$ where i denotes the identity of a cheater.

3.1.2 Security We require one-or-nothing secret sharing with intermediaries to satisfy *privacy* and *identifiability*, described below. Notice that identifiability naturally implies correctness. Our definitions of privacy and identifiability both assume that corrupt parties might provide honest parties, including the dealer, with inconsistent or incorrect public keys. Below, we denote the set of n parties as $\{D, P_1, \ldots, P_{n-1}\}$, where D denotes the dealer.

Informal Definition 1 (`1or0wi`: Privacy) *Informally, this property requires that when fewer than $n-2t$ honest parties produce their ballot using v, then the adversary learns nothing about $\mathtt{z}^{(v)}$.*

We refer to Definition 10 in Appendix B for a formal definition.

The one-or-nothing secret sharing of Damgård *et al.* [DMR+21] additionally required *contradiction-privacy.* This guaranteed the privacy of all secrets when a pair of honest parties produce ballots for different indices. Notably, our one-or-nothing secret sharing with intermediaries does not have this property; however, when $n > 3t$, the privacy property implies that at most one secret is reconstructed.[11]

Informal Definition 2 (`1or0wi`: Identifiability) *Informally, this property requires that when at least $n-t$ parties produce their ballot using the same v, either* `reconstruct` *returns $\mathtt{z}^{(v)}$ or a corrupt party is identified.*

We refer to Definition 11 in Appendix B for a formal definition. It is easy to see that the identifiability property defined above implies *correctness* (i.e. when all algorithms are executed honestly, if at least $n-t$ parties produce their ballot using the same v, `reconstruct` returns $\mathtt{z}^{(v)}$).

3.1.3 Construction

Both the one-or-nothing secret sharing scheme of Damgård *et al.* [DMR+21] and our construction of one-or-nothing secret sharing with intermediaries make use of two layers of Shamir secret sharing. However, Damgård *et al.* crucially differ in the way in which the sub-shares for reconstructing a given value are transferred by the shareholders. Because without a PKI a dealer might not communicate reliably/verifiably to all share recipients (as either she or they might be corrupt), in order to achieve identifiability in such scenarios, we introduce a new tool which we informally call *transferrable encryption.*

Transferrable encryption allows a sender to encrypt a message to an intermediary, who, even before seeing the ciphertext, can *transfer* the ability to decrypt to another receiver. This can be achieved, for instance, simply by having the intermediary encrypt her (single-use) secret decryption key to the receiver.

We now informally describe the one-or-nothing secret sharing with intermediaries algorithms `keygen`, `share`, `vote`, and `reconstruct`:

1. Informally, `keygen` generates many single-use public-key encryption key pairs for each party i, designated for transference of decryption power to different parties j. Each party i will end up with a key pair $(\mathtt{sk}^{(v)}_{j\to i}, \mathtt{pk}^{(v)}_{j\to i})$ for every party j and shared value index v.

[11] Suppose, for contradiction, secrets at two indices are reconstructed: let those indices be v and v'. Then, privacy dictates that at least $n-2t$ honest parties must have produced ballots for v and a (disjoint) set of $n-2t$ honest parties must have produced ballots for v'. This can occur only if $(n-2t)+(n-2t) \leq n-t$ holds, which contradicts the assumption that $n > 3t$.

2. In the $\texttt{share}$ algorithm the dealer threshold secret shares each secret $\mathtt{z}^{(v)}$ as $s_1^{(v)}, \ldots, s_n^{(v)}$, and then threshold secret shares each $s_i^{(v)}$ as $s_{i\to 1}^{(v)}, \ldots, s_{i\to n}^{(v)}$. Then, the dealer broadcasts an encryption of each sub-share $s_{i\to j}^{(v)}$ under a key $\mathtt{pk}_{i\to j}^{(v)}$ belonging to party j; later, during $\texttt{vote}$, party j will act as an intermediary, and forward that share to party i.
3. $\texttt{vote}$ is divided into two sub-steps (the first of which is independent of the party's vote):
   (a) Each party j broadcasts *transfer keys* for each index v and each other party i that can be applied to the encryption of $s_{i\to j}^{(v)}$ (under party j's public key $\mathtt{pk}_{i\to j}^{(v)}$) to make it decryptable using party i's secret decryption key $\mathtt{sk}_{i\to i}^{(v)}$. (Such a transfer key can simply be an encryption of $\mathtt{sk}_{i\to j}^{(v)}$ under party i's public key $\mathtt{pk}_{i\to i}^{(v)}$.)
   (b) To vote for the reconstruction of $\mathtt{z}^{(v)}$, each party i broadcasts her relevant secret decryption key $\mathtt{sk}_{i\to i}^{(v)}$.
4. Finally, the $\texttt{reconstruct}$ algorithm decrypts all the shares made available through the broadcast of the relevant decryption keys, and reconstructs $\mathtt{z}^{(v)}$ if at least $n-t$ votes supported v; otherwise, a cheating party is identified.

Finally, to achieve security against an active adversary, each party provides a non-interactive zero-knowledge proof (NIZK) to ensure that each step is honestly computed. Therefore, the $\texttt{setup}$ algorithm is also tasked with providing the CRSs required for the NIZKs.

More formally, let $\mathtt{PKE} = (\mathtt{keygen}, \mathtt{enc}, \mathtt{dec})$ be a public key encryption scheme with CPA security, and let $\mathtt{NIZK} = (\mathtt{setupZK}, \mathtt{prove}, \mathtt{verify}, \mathtt{simP}, \mathtt{extract})$ be a non-interactive zero-knowledge proof system for the following relations:

$$\mathcal{R}_{\mathtt{keygen}} = \left\{ \begin{array}{l} \phi = \mathtt{pk} \\ \\ w = (\mathtt{sk}, r) \end{array} \middle| (\mathtt{sk}, \mathtt{pk}) \leftarrow \mathtt{PKE.keygen}(1^\lambda; r) \right\},$$

$$\mathcal{R}_{\mathtt{share}} = \left\{ \begin{array}{l} \phi = \{\mathtt{pk}_{i\to j}^{(v)}, c_{i\to j}^{(v)}\}_{v\in[l], i,j\in[n]} \\ w = \left(\begin{array}{l} \{\mathtt{z}^{(v)}, r^{(v)}, \{r_i^{(v)}, \\ \{r_{i\to j}^{(v)}\}_{j\in[n]}\}_{i\in[n]}\}_{v\in[l]} \end{array} \right) \end{array} \middle| \begin{array}{l} \big\{(s_1^{(v)}, \ldots, s_n^{(v)}) \leftarrow \mathtt{Shamir.share}(\mathtt{z}^{(v)}; r^{(v)})\big\}_{v\in[l]} \\ \wedge \big\{(s_{i\to 1}^{(v)}, \ldots, s_{i\to n}^{(v)}) \leftarrow \mathtt{Shamir.share}(s_i^{(v)}; r_i^{(v)})\big\}_{v\in[l], i\in[n]} \\ \wedge \big\{c_{i\to j}^{(v)} \leftarrow \mathtt{PKE.enc}(\mathtt{pk}_{i\to j}^{(v)}, s_{i\to j}^{(v)}; r_{i\to j}^{(v)})\big\}_{v\in[l], i,j\in[n]} \end{array} \right\},$$

$$\mathcal{R}_{\mathtt{vote}} = \left\{ \begin{array}{l} \phi = \left(\begin{array}{l} \{\mathtt{pk}_{j\to j}^{(v)}, \mathtt{pk}_{j\to i}^{(v)}, \mathtt{tk}_{j\to i}^{(v)}\}_{v\in[l], j\in[n]}, \\ v_i, \mathtt{sk}_{i\to i}^{(v_i)} \end{array} \right) \\ w = \left(\{\mathtt{sk}_{j\to i}^{(v)}, \bar{r}_{j\to i}^{(v)}, r_j^{(v)}\}_{v\in[l], j\in[n]} \right) \end{array} \middle| \begin{array}{l} \big\{(\mathtt{sk}_{j\to i}^{(v)}, \mathtt{pk}_{j\to i}^{(v)}) \leftarrow \mathtt{PKE.keygen}(1^\lambda; \bar{r}_{j\to i}^{(v)})\big\}_{j\in[n], v\in[l]} \\ \wedge \big\{\mathtt{tk}_{j\to i}^{(v)} \leftarrow \mathtt{PKE.enc}(\mathtt{pk}_{j\to j}^{(v)}, \mathtt{sk}_{j\to i}^{(v)}; r_j^{(v)})\big\}_{v\in[l], j\in[n]} \end{array} \right\}.$$

Figure 3.1 describes our one-or-nothing secret sharing with intermediaries ($\mathtt{1or0wi}$) scheme.

Figure 3.1: Construction of `1or0wi`

$\mathtt{setup}(1^\lambda)$: Set up and output the common reference strings

$crs_{\mathtt{keygen}} \leftarrow \mathtt{setupZK}(1^\lambda, \mathcal{R}_{\mathtt{keygen}})$,

$crs_{\mathtt{share}} \leftarrow \mathtt{setupZK}(1^\lambda, \mathcal{R}_{\mathtt{share}})$, and

$crs_{\mathtt{vote}} \leftarrow \mathtt{setupZK}(1^\lambda, \mathcal{R}_{\mathtt{vote}})$

for the zero knowledge proof system. Return $crs = (crs_{\mathtt{keygen}}, crs_{\mathtt{share}}, crs_{\mathtt{vote}})$.

$\mathtt{keygen}(crs)$**, run by party** i**:**

1. For each $j \in [n]$ and $v \in [l]$, $(\mathtt{sk}^{(v)}_{j\to i}, \mathtt{pk}^{(v)}_{j\to i}) \leftarrow \mathtt{PKE.keygen}(1^\lambda; \bar{r}^{(v)}_{j\to i})$.
2. For each $j \in [n]$ and $v \in [l]$, $\pi^{(v)}_{j\to i} \leftarrow \mathtt{NIZK.prove}(crs_{\mathtt{keygen}}, \phi = \mathtt{pk}^{(v)}_{j\to i}, w = (\mathtt{sk}^{(v)}_{j\to i}, \bar{r}^{(v)}_{j\to i}))$.
3. Let $\mathtt{sk}_i = (\{\mathtt{sk}^{(v)}_{j\to i}, \bar{r}^{(v)}_{j\to i}\}_{j\in[n], v\in[l]})$, and $\mathtt{pk}_i = (\{\mathtt{pk}^{(v)}_{j\to i}, \pi^{(v)}_{j\to i}\}_{j\in[n], v\in[l]})$.
4. Output $(\mathtt{sk}_i, \mathtt{pk}_i)$.

$\mathtt{share}(crs, \mathtt{pk}_1, \ldots, \mathtt{pk}_n, \mathtt{z}^{(1)}, \ldots, \mathtt{z}^{(l)})$**, run by the dealer** D **(where** $\mathtt{pk}_i = \{\mathtt{pk}^{(v)}_{j\to i}, \pi^{(v)}_{j\to i}\}_{j\in[n], v\in[l]}$**):**

1. For each $v \in [l]$, compute $(s^{(v)}_1, \ldots, s^{(v)}_n) \leftarrow \mathtt{Shamir.share}(\mathtt{z}^{(v)}; r^{(v)})$ as the threshold sharing of $\mathtt{z}^{(v)}$ with threshold $(n - t - 1)$.
2. For each $i \in [n]$ and $v \in [l]$, compute $(s^{(v)}_{i\to 1}, \ldots, s^{(v)}_{i\to n}) \leftarrow \mathtt{Shamir.share}(s^{(v)}_i; r^{(v)}_i)$ as the threshold sharing of $s^{(v)}_i$ with threshold $(n - 2t - 1)$.
3. For each $i, j \in [n]$ and $v \in [l]$, compute $c^{(v)}_{i\to j} \leftarrow \mathtt{PKE.enc}(\mathtt{pk}^{(v)}_{i\to j}, s^{(v)}_{i\to j}; r^{(v)}_{i\to j})$.
4. Set
   - $\phi_{\mathtt{share}} = (\{\mathtt{pk}^{(v)}_{i\to j}, c^{(v)}_{i\to j}\}_{v\in[l], i,j\in[n]})$ and
   - $w_{\mathtt{share}} = (\{\mathtt{z}^{(v)}, r^{(v)}, \{r^{(v)}_i, \{r^{(v)}_{i\to j}\}_{j\in[n]}\}_{i\in[n]}\}_{v\in[l]})$.

   Compute $\pi_{\mathtt{share}} \leftarrow \mathtt{prove}(crs_{\mathtt{share}}, \phi_{\mathtt{share}}, w_{\mathtt{share}})$.
5. Set $s = (\phi_{\mathtt{share}}, \pi_{\mathtt{share}})$ and output s.

$\mathtt{vote}(crs, \mathtt{sk}_i, \mathtt{pk}_1, \ldots, \mathtt{pk}_n, v_i)$**, run by party** i **(where** $\mathtt{pk}_i = \{\mathtt{pk}^{(v)}_{j\to i}, \pi^{(v)}_{j\to i}\}_{j\in[n], v\in[l]}$ **and** $\mathtt{sk}_i = \{\mathtt{sk}^{(v)}_{j\to i}, \bar{r}^{(v)}_{j\to i}\}_{j\in[n], v\in[l]}$**):**

1. For each $v \in [l]$ and $j \in [n]$, let $\mathtt{tk}^{(v)}_{j\to i} \leftarrow \mathtt{PKE.enc}(\mathtt{pk}^{(v)}_{j\to j}, \mathtt{sk}^{(v)}_{j\to i}; r^{(v)}_j)$.
2. Set
   - $\phi_{\mathtt{vote},i} = (\{\mathtt{pk}^{(v)}_{j\to j}, \mathtt{pk}^{(v)}_{j\to i}, \mathtt{tk}^{(v)}_{j\to i}\}_{v\in[l], j\in[n]}, v_i, \mathtt{sk}^{(v_i)}_{i\to i})$ [a]
   - $w_{\mathtt{vote},i} = (\{\mathtt{sk}^{(v)}_{j\to i}, \bar{r}^{(v)}_{j\to i}, r^{(v)}_j\}_{v\in[l], j\in[n]})$.

   Compute $\pi_{\mathtt{vote},i} \leftarrow \mathtt{prove}(crs_{\mathtt{vote}}, \phi_{\mathtt{vote},i}, w_{\mathtt{vote},i})$.
3. Set $\overline{s}_i = (\phi_{\mathtt{vote},i}, \pi_{\mathtt{vote},i})$ and output $\overline{s}_i$.

$\texttt{reconstruct}(crs, s, (\texttt{pk}_1, v_1, \overline{s}_1), \ldots, (\texttt{pk}_n, v_n, \overline{s}_n))$ **(where** $s = (\{\texttt{pk}^{(v)}_{i\to j}, c^{(v)}_{i\to j}\}_{v\in[l],i,j\in[n]}, \pi_{\texttt{share}})$**,** $\texttt{pk}_i = \{\texttt{pk}^{(v)}_{j\to i}, \pi^{(v)}_{j\to i}\}_{j\in[n],v\in[l]}$ **and** $\overline{s}_i = (\phi_{\texttt{vote},i} = (\{\texttt{pk}^{(v)}_{j\to j}, \texttt{pk}^{(v)}_{j\to i}, \texttt{tk}^{(v)}_{j\to i}\}_{v\in[l],j\in[n]}, v_i, \texttt{sk}^{(v_i)}_{i\to i}), \pi_{\texttt{vote},i})$**):**

Identify the winning vote:

1. If there does not exist a $v \in \{1, \ldots, l\}$ such that at least $(n-t)$ parties vote for v, output $\perp$. Let $S_{\texttt{vote}} \subseteq [n]$ be the set of parties i such that $v_i = v$.

Verify the zero knowledge proofs:

2. For $i, j \in [n]$, if $\texttt{NIZK.verify}(crs_{\texttt{keygen}}, \phi = \texttt{pk}^{(v)}_{j\to i}, \pi^{(v)}_{j\to i}) = \texttt{reject}$ (where $\texttt{pk}^{(v)}_{j\to i}$, $\pi^{(v)}_{j\to i}$ are taken from $\texttt{pk}_i$), return $\perp_i$.
3. If $\texttt{NIZK.verify}(crs_{\texttt{share}}, \phi_{\texttt{share}}, \pi_{\texttt{share}}) = \texttt{reject}$ (where $\phi_{\texttt{share}}$, $\pi_{\texttt{share}}$ are taken from s), return $\perp_D$.
4. For $i \in [n]$, if $\texttt{NIZK.verify}(crs_{\texttt{vote}}, \phi_{\texttt{vote},i}, \pi_{\texttt{vote},i}) = \texttt{reject}$ (where $\phi_{\texttt{vote},i}$, $\pi_{\texttt{vote},i}$ are taken from $\overline{s}_i$), return $\perp_i$.

Check the consistency of the share, ballots and keys:

5. For $i \in [n]$, let $S'_i \subseteq [n]$ be the set of parties $j \in [n]$ such that (a) $\texttt{pk}^{(v)}_{i\to i}$ is the same in $\texttt{pk}_i$ and $\overline{s}_j$, and (b) $\texttt{pk}^{(v)}_{j\to j}$ is the same in $\texttt{pk}_j$ and $\overline{s}_i$. If $|S'_i| < n - t$, return $\perp_i$.
6. Let $S_D \subseteq [n]$ be the set of parties i such that $\{\texttt{pk}^{(v)}_{j\to i}\}_{j\in[n]}$ is the same in $\texttt{pk}_i$ and s. If $|S_D| < n - t$, return $\perp_D$.
7. For $i \in S_{\texttt{vote}}$, let $S_i = S'_i \cap S_D$. Note that $|S_i| \geq n - 2t$.
   For $j \in S_i$, we have a ciphertext $\texttt{tk}^{(v)}_{i\to j}$ (contained in $\overline{s}_j$), a secret key $\texttt{sk}^{(v)}_{i\to i}$ (contained in $\overline{s}_i$) and a ciphertext $c^{(v)}_{i\to j}$ (contained in s). Let $\texttt{sk}^{(v)}_{i\to j} \leftarrow \texttt{PKE.dec}(\texttt{sk}^{(v)}_{i\to i}, \texttt{tk}^{(v)}_{i\to j})$. Let $s^{(v)}_{i\to j} \leftarrow \texttt{PKE.dec}(\texttt{sk}^{(v)}_{i\to j}, c^{(v)}_{i\to j})$.
8. For each $i \in S_{\texttt{vote}}$, let $s^{(v)}_i \leftarrow \texttt{Shamir.reconstruct}(\{s^{(v)}_{i\to j}\}_{j\in S_i})$.
9. Output $\texttt{z}^{(v)} \leftarrow \texttt{Shamir.reconstruct}(\{s^{(v)}_i\}_{i\in S_{\texttt{vote}}})$.

Maliciously secure one-or-nothing secret sharing with intermediaries when $n > 3t$.

[a] any string $m^{(\perp)}$ is to be interpreted as $\perp$.

Theorem 1. *The construction in Figure 3.1 is a maliciously secure one-or-nothing secret sharing with intermediaries when $n > 3t$ if* PKE *is a public key encryption scheme with CPA security, and* NIZK *is a secure non-interactive zero-knowledge proof system.*

We prove Theorem 1 in Appendix C.

3.2 IA Feasibility Result: P2P-BC, IA, $3t < n$

Our upper bounds are based on those of Cohen *et al.* [CGZ20] and Damgård *et al.* [DMR+21]. They take a BC-BC protocol $\Pi_{\texttt{bc}}$, and *compile* it to the P2P-BC setting. The primary challenge here is making sure that corrupt parties cannot break security by sending different messages to honest parties in the first round. Our compiler makes sure that if corrupt party first-round messages are *consistent enough*, honest party second-round messages are produced on the same set of first-round messages; otherwise, a corrupt party is unanimously identified. To achieve this, we (and the prior works) have each party garble her second-message function, which has her own input hardcoded, and takes as input all the first-round messages she receives. Each party also secret-shares all of the labels for her own garbled circuit. In the second round, over broadcast, parties echo the first-round messages they received, distribute their garbled circuit, and contribute to label reconstruction (for everyone's garbled circuits) corresponding to the first-round messages they received. If there aren't $n-t$ parties who all echo the same first-round message from a given P_i, honest parties abort blaming P_i; if there aren't $n-t$ parties who all contribute valid ballots for P_j's labels, honest parties abort blaming P_j. Note that if an (identifiable) abort happens, reconstruction is allowed to fail.

Using Shamir secret sharing with threshold $s = \frac{3n}{5}$, this leads to a P2P-BC, IA, CRS protocol with $t < \frac{n}{5}$. The reason we have corruption threshold $t = \frac{n}{5}$ and sharing threshold $s = \frac{3n}{5}$ is that we have two constraints:

1. In order to prevent the adversary from learning two labels for the same wire by sending different first-round messages to two subsets of the honest parties, we need $s \geq t + \frac{n-t}{2}$.
2. In order to ensure that even after (a) t parties echo a different message from party m and (b) a different t parties give bad label shares we still have enough shares to reconstruct, we need $s < n-2t$. (If only t parties have inconsistent claims with the message sender and a different t parties have inconsistent claims with the label share dealer, we have no idea who to blame, so we *have to* reconstruct!)

We get

$$\begin{aligned} t + \frac{n-t}{2} &\leq s < n - 2t \\ \Rightarrow \qquad t + n &< 2n - 4t \\ \Rightarrow \qquad 5t &< n. \end{aligned}$$

However, $5t < n$ does not match the lower bound from Theorem 4.

To match the lower bound we need a more sophisticated mechanism of sharing such that *all* parties can contribute valid shares of each label, or someone is unanimously identified as a cheater. In Section 3.1 we construct exactly such a primitive, which we call one-or-nothing secret sharing with intermediaries (`1or0wi`). Intuitively, our one-or-nothing secret sharing with intermediaries achieves this goal by having each dealer use all of the parties as intermediaries to all share recipients; if sufficiently many intermediaries don't succeed in helping the dealer G give a share to a recipient P, then either the dealer or the recipient can be identified as corrupt, since they are in conflict with more than t intermediaries (we refer to Section 3.1 to a more detailed description of how this works).

We are now ready to describe our final protocol with identifiable abort for threshold $3t < n$. In the first round (which is over public peer-to-peer channels), the parties send their first-round messages of $\Pi_{\mathtt{bc}}$ along with the public keys produced by the key generation algorithm of `1or0wi`. In the second round (which is over broadcast), the parties execute the following steps:

1. They compute a garbling of the second-message function of $\Pi_{\mathtt{bc}}$;
2. they use `1or0wi` to share the labels of their garbled circuit;
3. they use `1or0wi` to vote for the labels of the garbled circuits of the other participants based on the first-round messages of $\Pi_{\mathtt{bc}}$ (received in the peer-to-peer round); and
4. they echo the first-round messages of $\Pi_{\mathtt{bc}}$ received in the first round.

Before computing the output, each party P_i performs some validations on the echoed messages. Namely, P_i checks that (a) all the parties generated their ballots for each garbled circuit based on the first-round messages that they echoed, and (b) all the parties have mutual successful communication with at least $n - t$ others in the first round. If there is a party P_j that does not pass these checks, party P_i identifies P_j as a cheater. If all of the parties pass the checks, then party P_i invokes the `reconstruct` algorithm of `1or0wi`. If `reconstruct` blames party P_j, P_i aborts and identifies that party as a cheater. Otherwise, P_i reconstructs labels for all the garbled circuits, uses the garbled circuits to obtain the second-round messages of $\Pi_{\mathtt{bc}}$, and uses those second-round messages to complete the protocol and obtain the computation output.

Roughly speaking, the identifiable abort property is guaranteed since the one-or-nothing secret sharing with intermediaries is secure against active adversaries. Therefore, if the two validations (a) and (b) succeed, we can rely on the properties of `1or0wi` to guarantee that $\Pi_{\mathtt{bc}}$ is executed or a malicious party is identified.

More formally our protocol is described in Figure 3.2 and we assume that the parties have access to the following tools:

Tools.

- A BC-BC, IA, CRS protocol i.e. a two-round broadcast protocol $\Pi_{\mathtt{bc}}$ achieving security with identifiable abort. (This could, for instance, be the protocol described by Cohen *et al.* [CGZ20].)
  $\Pi_{\mathtt{bc}}$ is 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 ($\mathtt{garble}, \mathtt{eval}, \mathtt{simGC}$) (defined in Appendix A.4).
- A one-or-nothing secret sharing with intermediaries
  $\mathtt{1or0wi} = (\mathtt{setup}, \mathtt{keygen}, \mathtt{share}, \mathtt{vote}, \mathtt{reconstruct})$ (defined in Section 3.1).

Notation. Let $\mathsf{C}_i(x_i, r_i, \mathsf{msg}_1^1, \ldots, \mathsf{msg}_n^1)$ denote the boolean circuit that takes P_i's input x_i, randomness r_i and the first round messages $\mathsf{msg}_1^1, \ldots, \mathsf{msg}_n^1$, and outputs msg_i^2. For simplicity assume that (x_i, r_i) consists of z bits, and each first round message is ℓ bits long, so each circuit has $\mathsf{L} = z + n \cdot \ell$ input bits. Note that C_i is public. Let g be the size of a garbled C_i.

Figure 3.2: $\Pi^{\mathsf{id\text{-}abort}}_{\mathsf{p2pbc}}$ with $n > 3t$

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

Setup.

- CRS setup for one-or-nothing secret sharing with intermediaries: $crs \leftarrow \mathtt{setup}(1^\lambda)$.
- Setup for Π_{bc} (which includes CRS when instantiated using the protocol of [CGZ20]). [a]

First Round. Each party P_i does the following:

1. Let $(\mathtt{sk}_i, \mathtt{pk}_i) \leftarrow \mathtt{keygen}(1^\lambda)$, where $\mathtt{pk}_i = \big\{\mathtt{pk}_i^{(1)} = (\mathtt{pk}_i^{(1,1)}, \ldots, \mathtt{pk}_i^{(1,\mathsf{L})}), \ldots, \mathtt{pk}_i^{(n)} = (\mathtt{pk}_i^{(n,1)}, \ldots, \mathtt{pk}_i^{(n,\mathsf{L})})\big\}$ is a vector of $n\mathsf{L}$ public keys with the corresponding vector of secret keys $\mathtt{sk}_i = \big\{\mathtt{sk}_i^{(1)} = (\mathtt{sk}_i^{(1,1)}, \ldots, \mathtt{sk}_i^{(1,\mathsf{L})}), \ldots, \mathtt{sk}_i^{(n)} = (\mathtt{sk}_i^{(n,1)}, \ldots, \mathtt{sk}_i^{(n,\mathsf{L})})\big\}$ (We abuse notation slightly by assuming that $\mathtt{keygen}(1^\lambda)$ outputs a vector of public keys and secret keys; we do this for simplicity)
2. Let $\mathsf{msg}_i^1 \leftarrow \mathtt{frst\text{-}msg}_i(x_i, r_i)$ be P_i's first round message in Π_{bc}.
3. Send $(\mathtt{pk}_i, \mathsf{msg}_i^1)$ to P_j for $j \in [n]$.

Second Round. Each party P_i does the following:

We specify multiple broadcast messages separately for clarity; however, they are all sent simultaneously as a single round of communication.

1. Let $\mathtt{pk}_{j\to i} = \{\mathtt{pk}_{j\to i}^{(1)}, \ldots, \mathtt{pk}_{j\to i}^{(n)}\}$ denote the $\mathtt{pk}_j$ received privately from P_j ($j \in [n]$), where $\mathtt{pk}_{j\to i}^{(k)} = (\mathtt{pk}_{j\to i}^{(k,1)}, \ldots, \mathtt{pk}_{j\to i}^{(k,\mathsf{L})})$ for $k \in [n]$.
2. Compute $(\mathtt{GC}_i, \vec{K}_i) \leftarrow \mathtt{garble}(1^\lambda, \mathtt{C}_i; R_i)$, where $\vec{K}_i = \{K_{i,l}^{(0)}, K_{i,l}^{(1)}\}_{l\in[\mathsf{L}]}$.
3. For every $l \in [z+1, \ldots, \mathsf{L}]$, let $s_{i,l} \leftarrow \mathtt{share}(crs, \mathtt{pk}_{1\to i}^{(i,l)}, \ldots, \mathtt{pk}_{n\to i}^{(i,l)}, K_{i,l}^{(0)}, K_{i,l}^{(1)})$. Broadcast $\{s_{i,l}\}_{l\in[z+1,\ldots,\mathsf{L}]}$.
4. Let $(\nu_{i,z+1}, \ldots, \nu_{i,\mathsf{L}})$ denote the bits comprising $(\mathsf{msg}_{1\to i}^1, \ldots, \mathsf{msg}_{n\to i}^1)$, where $\mathsf{msg}_{j\to i}^1$ refers to msg_j^1 received from P_j in Round 1.
5. For each $k \in [n]$ and $l \in [z+1, \mathsf{L}]$: Compute and broadcast $\overline{s}_{i,l}^{(k)} \leftarrow \mathtt{vote}(crs, \mathtt{sk}_i^{(k,l)}, \mathtt{pk}_{1\to i}^{(k,l)}, \ldots, \mathtt{pk}_{n\to i}^{(k,l)}, \nu_{i,l})$.

   Broadcast own garbled circuit:
6. Let $(\nu_{i,1}, \ldots, \nu_{i,z})$ denote the bits corresponding to (x_i, r_i).
7. For $l \in [z]$, let $K_{i,l} = K_{i,l}^{(\nu_{i,l})}$.
8. Broadcast $(\mathtt{GC}_i, \{K_{i,l}\}_{l\in[z]})$.

   Echo first-round messages:
9. Broadcast $(\mathsf{msg}_{1\to i}^1, \ldots, \mathsf{msg}_{n\to i}^1)$.

   Let $\mathsf{msg}_i^1 = \mathsf{msg}_{i\to i}^1$ denote the party's own first-round message.

Output Computation. Each party P_i does the following:

If there is a party who did not generate ballots for each garbled circuit based on the first-round messages that she echoed, blame that party:

1. For $j \in [n]$: Check if $\{\mathsf{msg}^1_{k\to j}\}_{k\in[n]}$ broadcast by P_j is consistent with $\{\overline{s}^{(k)}_{j,l}\}_{k\in[n],l\in[z+1,\mathsf{L}]}$ [b]. Output $\mathtt{abort}_j$ if the check fails. Else, set $(\nu_{j,z+1},\ldots,\nu_{j,\mathsf{L}})$ as the bits comprising $(\mathsf{msg}^1_{1\to j},\ldots,\mathsf{msg}^1_{n\to j})$.

   If there is a party who did not have mutual successful communication with at least $n-t$ others in the first round, blame that party:
2. For $j \in [n]$: If there does not exist a set $|S_j| \geq n-t$ such that, for $k \in S_j$, $\mathsf{msg}^1_{j\to k} = \mathsf{msg}^1_j$ holds; output $\mathtt{abort}_j$.

   Decrypt the shares:
3. For $k \in [n]$ (whose garbled circuit we will now consider):
   (a) For $l \in [z+1,\mathsf{L}]$, compute $K_{k,l} \leftarrow \mathtt{reconstruct}(crs, s_{k,l}, (\mathtt{pk}^{(k,l)}_1, v_{1,l}, \overline{s}^{(k)}_{1,l}),\ldots,(\mathtt{pk}^{(k,l)}_n, v_{n,l}, \overline{s}^{(k,l)}_{n,l})$. If $\mathtt{reconstruct}$ returns $\bot_{\mathtt{id}}$, output $\mathtt{abort}_{\mathtt{id}}$. Else, continue.
   (b) Evaluate $\mathsf{msg}^2_k \leftarrow \mathtt{eval}(\mathtt{GC}_k,(K_{k,1},\ldots,K_{k,\mathsf{L}}))$. If the evaluation fails, output $\mathtt{abort}_k$.
4. Output $y \leftarrow \mathtt{output}_i(x_i, r_i, \mathsf{msg}^1_1,\ldots,\mathsf{msg}^1_n,\mathsf{msg}^2_1,\ldots,\mathsf{msg}^2_n)$.

P2P-BC, IA, $t < \frac{n}{3}$ secure computation in the CRS model.

[a]For simplicity (to avoid introducing additional notation), we assume implicitly that the set of functions $\{\mathtt{frst\text{-}msg}_i, \mathtt{snd\text{-}msg}_i, \mathtt{output}_i\}_{i\in[n]}$ of Π_{bc} use the relevant setup information.

[b]Note that in our construction of one-or-nothing secret sharing with intermediaries, it is possible to retrieve the corresponding vote directly from the ballot $\overline{s}^{(k)}_{j,l}$.

Theorem 2 (P2P-BC, ID, CRS, $n > 3t$). *Let f be an efficiently computable n-party function and let $n > 3t$. Let Π_{bc} be a BC-BC, ID, CRS protocol that securely computes f with the additional constraint that the straight-line simulator can extract inputs from corrupt parties' first-round messages. Assuming that* $(\mathtt{garble}, \mathtt{eval}, \mathtt{simGC})$ *is a secure garbling scheme, and* $(\mathtt{setup}, \mathtt{keygen}, \mathtt{share}, \mathtt{vote}, \mathtt{reconstruct})$ *is a secure one-or-nothing secret sharing with intermediaries. Then, $\Pi^{\mathsf{id\text{-}abort}}_{\mathsf{p2pbc}}$ securely computes 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 and peer-to-peer channels.*

We prove Theorem 2 in Appendix D.

3.3 Feasibility results for SIA

Our positive results for SIA rely on the following theorem.

Theorem 3. *Let Π_{bc} be a BC-BC protocol (respectively a P2P-BC) that securely computes f with identifiable abort security against t corruptions with the additional properties that the simulator can extract inputs from the first-round* *messages and it is efficient to check whether a given second-round message is correct. Then $\Pi_{\mathtt{bc}}$ securely computes f with selective identifiable-abort security against t corruptions when the second round is run over peer-to-peer channels instead.*

We prove Theorem 3 in Appendix E, where we also argue that the IA constructions of [CGZ20], [DMR$^+$21] and Theorem 2 satisfy the above stated properties (and can therefore be compiled to yield P2P-P2P, SIA, CRS, $t < \frac{n}{3}$ protocol, BC-P2P, SIA, CRS, $t < n$ protocol and P2P-P2P, SIA, PKI, $t < \frac{n}{2}$) protocol.

4 Lower Bounds

4.1 General Impossibility Results

In this section, we present impossibility results that hold even when the peer-to-peer channels in the first round are private.

Theorem 4 (P2P-BC, SIA, CRS, $n \leq 3t$). *There exist functions f such that no n-party two-round protocol can compute f with selective identifiable abort against $t \geq \frac{n}{3}$ corruptions while making use of broadcast only in the second round (i.e. where the first round is over peer-to-peer channels [12] and second round uses both broadcast and peer-to-peer channels).*

In our proof, we use the function $f_{\mathtt{ot}}$. Let the input of P_1, P_2 be a pair of strings $x_1 = (\mathsf{z}_0, \mathsf{z}_1)$, $x_2 = (\mathsf{z}'_0, \mathsf{z}'_1)$ where $\mathsf{z}_0, \mathsf{z}_1, \mathsf{z}'_0, \mathsf{z}'_1 \in \{0,1\}^\lambda$, and the input of P_n be a choice bit $x_n = c \in \{0,1\}$. The input of other parties is $\perp$ (i.e. $x_i = \perp$ for $i \in [n] \setminus \{1, 2, n\}$). $f_{\mathtt{ot}}$ allows everyone to learn $(\mathsf{z}_c, \mathsf{z}'_c)$.

Proof. We prove Theorem 4 by contradiction. Let Π be an n-party protocol computing $f_{\mathtt{ot}}$ that achieves identifiable abort against $t \geq \frac{n}{3}$ corruptions by using broadcast in the second round only.

For simplicity, we assume $n = 3$ and $t = 1$. We analyze the following scenarios in an execution of Π.

Scenario 1: The adversary does the following on behalf of P_3.

Round 1. Compute and send messages based on input $x_3 = 0$ and $x_3 = 1$ to P_1 and P_2 respectively. (It is possible for the adversary to send inconsistent first-round messages as the first round is communicated over peer-to-peer channels.)

Round 2. Discard the first-round message from P_2 and send messages based on input $x_3 = 0$. In other words, P_3 pretends as if she behaved honestly using input $x_3 = 0$ and did not receive a peer-to-peer message from P_2 in the first round.

Scenario 2: Consider an adversary who corrupts P_2. Suppose the input of honest P_3 is $x_3 = 0$. The adversary behaves as follows on behalf of P_2:

[12] The peer-to-peer channels can be private or "open".

Round 1. Behave honestly as per protocol specifications, except that the peer-to-peer message to P_3 is not sent.
Round 2. Pretend to have received first round messages from P_3 based on $x_3 = 1$. In more detail, the adversary drops the first round peer-to-peer message received from P_3 and replaces it by locally computing P_3's first round message based on input $x_3 = 1$ and some randomness (that the adversary can sample locally on behalf of P_3). Note that the adversary can do this without being caught, due to the absence of PKI or correlated randomness.

Claim. Π is such that P_1 in Scenario 1 learns the output $(\mathsf{z}_0, \mathsf{z}'_0)$ with all but negligible probability.

Proof. First, we observe that the view of honest P_1 in Scenario 1 is distributed identically to her view in Scenario 2. This is because in both scenarios, P_1 observes the following conflict between P_2 and P_3: P_3 claims to have not received the first-round peer-to-peer message from P_2 while P_2 claims to have received first-round peer-to-peer message from P_3 based on $x_3 = 1$. Therefore, to satisfy the guarantees of SIA, it must hold that either P_1 aborts in both scenarios or obtains the output in both scenarios. The former is not possible, since P_1 would identify the same cheater in both scenarios, which means that she would identify an honest party in one of the two scenarios (as the corrupt party is different in the two scenarios). We can thus infer from selective identifiable abort security guarantee of Π that both the above scenarios result in P_1 receiving an output, with all but negligible probability.

The output obtained by P_1 in Scenario 2 must include z_0 as it should be computed with respect to the input $x_3 = 0$ of honest P_3 and input $(\mathsf{z}_0, \mathsf{z}_1)$ of honest P_1. Therefore, the output obtained by P_1 in Scenario 1 should also include z_0 (with all but negligible probability). Infact, we can argue that the output obtained by P_1 in Scenario 1 should in fact be $(\mathsf{z}_0, \mathsf{z}'_0)$ (with all but negligible probability) to be consistent with the ideal realization of f. This is because the simulator in Scenario 1 can induce an output comprising of z_0 only by invoking the ideal functionality with $x_3 = 0$ on behalf of corrupt P_3, which fixes the output of P_1 to include z'_0 as per the definition of f.

We can thus conclude that the output obtained by honest P_1 in Scenario 1 must be $(\mathsf{z}_0, \mathsf{z}'_0)$. Next, we consider another Scenario, say **Scenario 3** –

Scenario 3: Adversary corrupts P_1 but behaves honestly throughout the protocol. Suppose the input of honest P_3 is $x_3 = 1$.

First, it follows from the correctness of the protocol that since all parties including the corrupt parties behaved honestly in Scenario 3, the output computed must be in fact $(\mathsf{z}_1, \mathsf{z}'_1)$ computed on honest inputs, which is obtained by all (including the adversary). Next, we show an attack by the adversary controlling P_1 that allows her to obtain z'_0 as well, which violates security (as an adversary corrupting P_1 is not allowed to learn both inputs of honest P_2 i.e. z'_0 and z'_1, as per the ideal computation of f). The main idea is that the adversary simulates *in her head* Scenario 1, where there was a conflict between P_3 and P_2.

In the above execution of Scenario 3, let $m_{i\to j}$ denotes the peer-to-peer first-round message sent by P_i to P_j and b_i denotes the second-round broadcast message sent by P_i (it is without loss of generality to assume that the second-round messages are over broadcast; since private communication in the second round can be realized by exchanging public keys in the first round).

Round 1: On behalf of P_3, the adversary chooses input $x_3 = 0$ and some chosen randomness, say r_3. Using these values, the adversary recomputes the outgoing first-round peer-to-peer message from P_3 to P_1, say $\overline{m_{3\to 1}}$. However, the other first-round peer-to-peer messages i.e. $m_{3\to 2}, m_{2\to 3}, m_{2\to 1}, m_{1\to 2}$ and $m_{1\to 3}$ are fixed to be the same as what were received during the execution of Scenario 3.

Round 2: Next, the adversary recomputes the second-round broadcast message of P_3, say $\overline{b_3}$ as follows: Compute the broadcast message based on protocol specifications when P_3 did not receive any first-round peer-to-peer message from P_2. Note that this message can be computed using input $x_3 = 0$, randomness r_3 and the first-round peer-to-peer message $m_{1\to 3}$ received by P_3 from P_1 (which the adversary knows). The broadcast message of P_1, say $\overline{b_1}$ is recomputed based on honest input and randomness of P_1, the above simulated first-round peer-to-peer message $\overline{m_{3\to 1}}$ and $m_{2\to 1}$. Lastly, the broadcast message of P_2 is fixed to b_2 (same as received in the execution).

We observe that the above simulation in her head, allows the adversary to obtain a view that is identically distributed to the view of honest P_1 in Scenario 1. This is because both the simulation as well as Scenario 1 involve the messages $\overline{m_{3\to 1}}$, $\overline{b_1}$ and $\overline{b_3}$ being based on $x_3 = 0$. We thus infer that the adversary should be able to compute the output of Scenario 1 as well.

Since the output of Scenario 1 is $(\mathtt{z}_0, \mathtt{z}'_0)$, we can conclude that the adversary of Scenario 3 learns both $\mathtt{z}'_0$ (via the simulation in her head) and $\mathtt{z}'_1$ (via the output of the execution) which violates security, since this is not allowed as per the ideal computation of f.

Lastly, we note that the above proof can be extended to $n \leq 3t$ using player partitioning technique (An n-party protocol Π' tolerating $t \geq n/3$ corruptions can be transformed into a 3-party protocol Π tolerating 1 corruption, by making a party in Π emulate the protocol steps of t parties in Π').

Theorem 5 (BC-P2P, UA, CRS, $t > 1$). *There exist functions f such that no n-party two-round protocol can compute f with unanimous abort against $t > 1$ corruptions while making use of broadcast only in the first round (i.e. where the first round uses both broadcast and peer-to-peer channels* [12] *and second round uses only peer-to-peer channels).*

In our proof, we use the function $f_{\mathtt{mot}}$. Let the input of P_n be a pair of strings $x_n = (\mathtt{z}_0, \mathtt{z}_1)$, where $\mathtt{z}_0, \mathtt{z}_1 \in \{0,1\}^\lambda$, and the input of every other party P_i ($i \in \{1, \ldots, n-1\}$) be a single bit $x_i \in \{0,1\}$. $f_{\mathtt{mot}}$ allows everyone to learn $\mathtt{z}_c$, where $c = \oplus_{i=1}^{n-1} x_i$.

Proof. We prove Theorem 5 by contradiction. Let Π be an n-party protocol securely computing $f_{\texttt{mot}}$ with unanimous abort using broadcast only in the first round. We describe a sequence of scenarios $H_1, C_2, H_2, \ldots, C_{n-1}, H_{n-1}, H_n$. The vector of inputs $(x_1, \ldots, x_n)$ is fixed across all scenarios. In all the scenarios, the adversary uses honest inputs for corrupt parties but may drop incoming or outgoing messages on behalf of corrupt parties.

We begin with a high-level overview of the argument. Each scenario H_i ($i \in \{2, \ldots, n\}$) involves the adversary corrupting P_1, while H_1 denotes an honest execution. The sequence of these scenarios is such that corrupt P_1 drops her peer-to-peer messages in both rounds to one additional honest party in each scenario. However, she behaves honestly in her first-round broadcast communication. The idea is to show that the output of H_i and H_{i+1} is identical for each $i \in [n-1]$. We show this by interleaving the above sequence with scenarios $C_2, \ldots, C_{n-1}$, in each of which P_1 as well as one other party are corrupt. (The only exception is C_2, where P_1 is honest.) This lets us infer that the output of Scenario H_n is identical to that of Scenario H_1: i.e. the output is $y = f(x_1, \ldots, x_n) = \mathsf{z}_c$, since in H_1 everyone behaves honestly and the output y is well-defined. Since the only communication from P_1 in Scenario H_n is her broadcast and private communication to P_n in Round 1, intuitively, P_n had sufficient information about x_1 required for output computation at the end of Round 1 itself. Building on this intuition, we demonstrate that Π is susceptible to a residual attack by P_n (which contradicts the security of Π).

Before describing the scenarios in detail, we first define some useful notation. Let μ denote the negligible probability with which the security of Π fails. Let $\widetilde{\mathsf{msg}}^2_{i \to j}$ denote P_i's second-round message to P_j (sent over peer-to-peer channel), computed honestly given that P_i did not receive the private message (i.e. the communication over peer-to-peer channel) from P_1 in the first round.

We now analyze the following sequence of scenarios $H_1, C_2, H_2, C_3, H_3, \ldots, C_{(n-1)}, H_{(n-1)}, H_n$.

Scenario H_1: All parties behave honestly. Since everyone behaved honestly, it follows from correctness that P_1 obtains the output $y = f(x_1, \ldots, x_n) = \mathsf{z}_c$ with probability at least $1 - \mu$.

Scenario C_2: P_2 is corrupt.

Round 1: P_2 behaves honestly.

Round 2: P_2 pretends to $P_3, \ldots, P_n$ that she did not receive private communication from P_1 in Round 1. In more detail, P_2 sends $\widetilde{\mathsf{msg}}^2_{2 \to j}$ to P_j for $j \in \{3, \ldots, n\}$.

P_1 must output $y = \mathsf{z}_c$ with probability at least $1 - \mu$ since her view is identically distributed to her view in Scenario H_1. Unanimity (which breaks with probability at most μ) dictates that when honest P_1 outputs $y = \mathsf{z}_c$ (which occurs with probability at least $1 - \mu$), the other honest parties $P_3, \ldots, P_n$ must also obtain $y = \mathsf{z}_c$ with probability at least $1 - 2\mu$.

Scenario H_2: P_1 is corrupt.

Round 1: P_1 behaves honestly in the broadcast communication and in private communication with $P_3, \dots, P_n$. However, she does not send private messages to P_2.

Round 2: P_1 communicates honestly with $P_3, \dots, P_n$ but sends no messages to P_2.

The views of parties $P_3, \dots, P_n$ is identically distributed to their views in the previous scenario (where the view of P_j, where $j \in \{3, \dots, n\}$ includes $\widetilde{\mathsf{msg}}^2_{2\to j}$). So, their output must be the same as in the previous scenario ($y = \mathsf{z}_c$) with probability at least $1 - 2\mu$. P_2 must also output $y = \mathsf{z}_c$ with probability at least $1 - 3\mu$ to maintain unanimity.

For $i \in [3, \dots, n-1]$, we define scenarios C_i and H_i as follows:

Scenario C_i: P_1 and P_i are corrupt.

Round 1: P_1 behaves as in the previous scenario (Scenario H_{i-1}). P_i behaves honestly.

Round 2: P_1 behaves as in the previous scenario. P_i behaves honestly towards $P_2, \dots, P_{i-1}$, and pretends to parties $P_{i+1}, \dots, P_n$ that she did not receive private communication from P_1 in round 1. In more detail, P_i sends $\widetilde{\mathsf{msg}}^2_{i\to j}$ to $j \in \{i+1, \dots, n\}$.

$P_2, \dots, P_{i-1}$ can't distinguish this from the previous scenario, since their views are identically distributed. So, their outputs must be the same as in the previous scenario (i.e. $y = \mathsf{z}_c$) with probability at least $1-(2(i-1)-1)\mu = 1-(2i-3)\mu$. $P_{i+1}, \dots, P_n$ must also output $y = \mathsf{z}_c$ with probability at least $1-(2i-3)\mu - \mu = 1-2(i-1)\mu$ to maintain unanimity.

Scenario H_i: P_1 is corrupt.

Round 1: P_1 behaves honestly in the broadcast communication and in private communication with $P_{i+1}, \dots, P_n$. However, she does not send private messages to $P_2, \dots, P_i$.

Round 2: P_1 communicates honestly with $P_{i+1}, \dots, P_n$ but sends no messages to $P_2, \dots, P_i$.

Parties $P_{i+1}, \dots, P_n$ can't distinguish this from the previous scenario, since their views are identically distributed (where the view P_j, where $j \in \{i+1, \dots, n\}$, includes $\{\widetilde{\mathsf{msg}}^2_{2\to j}, \dots, \widetilde{\mathsf{msg}}^2_{i\to j}\}$). So, their output must be the same as in the previous scenario (i.e. $y = \mathsf{z}_c$) with probability at least $1-2(i-1)\mu$. $P_2, \dots, P_i$ must also output $y = \mathsf{z}_c$ with probability at least $1-2(i-1)\mu-\mu = 1-(2i-1)\mu$ to maintain unanimity.

Finally, we define scenario H_n as follows:

Scenario H_n: P_1 is corrupt.

Round 1: P_1 behaves honestly in the broadcast communication and in private communication with P_n. However, she does not send private messages to $P_2, \dots, P_{n-1}$. (This is the same as in the previous — Scenario H_{n-1}.)

Round 2: P_1 sends no messages.

Parties $P_2, \ldots, P_{n-1}$ can't distinguish this from the previous scenario, since their views are identically distributed. So, their output must be the same as in the previous scenario (i.e. $y = \mathsf{z}_c$) with probability at least $1 - (2(n-1) - 1)\mu = 1 - (2n-3)\mu$. P_n must also output $y = \mathsf{z}_c$ with probability at least $1 - (2n-3)\mu - \mu = 1 - 2(n-1)\mu$ to maintain unanimity.

In the last scenario, we claim that the output (which is identical to z_c as shown above) is computed with respect to input $x'_1 = x_1$ of corrupt P_1. This is because the output in the last scenario must be computed on honest inputs of parties $P_2, \ldots, P_n$ (as they are honest in the last scenario) and $f(x'_1, x_2, \ldots, x_n) = \mathsf{z}_c$ holds only if $x'_1 = x_1$ (recall that $c = \oplus_{i=1}^{n-1} x_i$ is based on the fixed combination of inputs $(x_1, \ldots, x_n)$ used in the above scenarios).

Next, we note that in the last scenario, P_1 has spoken only in the first round broadcast and private communication to P_n. We argue that Π is susceptible to the following residual attack. Consider a different scenario H_n^* where the adversary passively corrupts P_n and behaves in the following manner. She fixes the first round broadcast and private communication obtained from P_1. She chooses inputs $(x'_2, \ldots, x'_n)$ and randomness on behalf of parties $P_2, \ldots P_n$. Now, it is easy to see that the adversary in H_n^* can locally emulate all the messages of Scenario H_n in her head with respect to inputs $(x_1, x'_2, x'_3, \ldots, x'_n)$. This is because the output of H_n was computed with respect to x_1 (as argued previously) and the only communication of P_1 throughout Scenario H_n was in its broadcast and private communication to P_n which is available to the adversary in H_n^* at the end of the first round. Lastly, since the adversary's view in H_n^* subsumes the view of honest P_n in Scenario H_n (if it occurred with respect to inputs $(x_1, x'_2 \ldots, x'_n)$), we can conclude that the adversary is able to learn the output $f(x_1, x'_2 \ldots, x'_n)$ with overwhelming probability. Thus, the adversary in H_n^* can obtain $f(x_1, x'_2 \ldots, x'_n)$ with overwhelming probability for any inputs $(x'_2, \ldots, x'_n)$ of her choice.

This contradicts the security of Π because it allows the adversary corrupting P_n to learn x_1. For instance, the adversary can choose $x'_n = (\mathsf{z}'_0, \mathsf{z}'_1)$ with $\mathsf{z}'_0 \neq \mathsf{z}'_1$ and $(x'_2, \ldots, x'_{n-1})$ such that $x'_2 \oplus x'_3 \oplus \cdots \oplus x'_{n-1} = 0$. This would allow her to deduce x_1 directly based on the output (which would evaluate to z'_{x_1}). This is in contrast to the ideal execution where the adversary passively corrupting P_n only learns c from the output which does not reveal x_1 (as c would depend on the inputs of other honest parties as well).

Theorem 5 extends the result of Patra and Ravi [PR18]. Our proof is quite different from theirs, though, as we need a sequence of significantly more hybrids to extend the result to the case when fewer than a third the participants are corrupt. Furthermore, we need to consider the simultaneous corruption of two (as opposed to only one among three parties considered in [PR18]) parties to argue the indistinguishability of honest parties' views in different scenarios.

Lastly, we point out that the above proof breaks down in the presence of a PKI or correlated randomness. This is because if such a setup is present, the adversary corrupting P_n in Scenario H_n^* will not be able to emulate the messages of an honest party P_i with respect to some chosen input x_i' (as the computation of these messages might require private information given to P_i as a part of the setup, which is not available to the adversary). In fact, there exist constructions achieving guaranteed output delivery (which implies unanimous abort) in the BC-P2P, PKI setting when $t < n/2$ [DMR+21].

4.2 Impossibility Results with Public First-Round Peer-to-Peer Channels

In this section, we present impossibility results for the setting where the peer-to-peer channels in the first round are public (observable by the adversary). [13] These results hold when the adversary corrupts one of n parties (where $n > 1$).

We begin with recalling some useful definitions and theorems from the work of Damgård *et al.* [DMR+21] (we refer to their paper for the formal details and proofs).

Definition 4 (Last Message Resiliency [DMR+21]). *A protocol is* r-last message resilient *if, in an honest execution, any protocol participant* P_i *can compute its output without using* r *of the messages it received in the last round.*

Theorem 6. *[DMR+21] 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, as long as* $n - t \geq 2$ *(that is, as long as there are at least two honest parties).*

Proof (Sketch). Informally, Damgård *et al.* argue that if Π is not t-last message resilient, then an adversary can disrupt unanimity between a pair of honest parties, say P_j and P_k, by doing the following: behaving honestly in the first round, behaving honestly in the second round towards P_k, and not sending messages on behalf of t corrupt parties to P_j. P_k will compute the correct output, since her view is the same as in an all-honest execution; however, P_j will not, since the protocol is not t-last message resilient.

Theorem 7. *[DMR+21] Any protocol* Π *which achieves secure computation with fairness with corruption threshold* t *must be* t*-last message resilient.*

Proof (Sketch). Informally, Damgård *et al.* argue that if Π is not t-last message resilient, then an adversary can violate fairness [14] by behaving honestly up until the last round and remaining silent in the last round (where the last round could use both broadcast and peer-to-peer channels). It is easy to see that the honest

[13] Recall that we assume throughout this work that the peer-to-peer channels in the second round are private, as this can be easily realized by parties sending their public keys in the first round which can then be used to encrypt messages in the second round.

[14] We assume that the function computed by Π is such that the adversary cannot compute the output locally by using her own inputs, therefore the argument for fairness can be invoked. This property holds for the function f that we consider later.

parties would not be able to compute the output if the protocol is not t-last message resilient, while the adversary would be, as her view subsumes the views of the corrupt parties in an all-honest execution.

Next, we show that two-round protocols using broadcast and public peer-to-peer channels in the first round cannot be t-last message resilient.

Theorem 8. *There exists a function f such that any two-round protocol Π securely realizing f whose first round can be executed over broadcast and public peer-to-peer channels cannot be r-last message resilient for $r > 0$. (Note that this holds regardless of of corruption threshold t; in particular, it holds even when $t = 0$, and all the adversary can do is observe the network.)*

Proof. We prove Theorem 8 by contradiction. Consider some function f where a residual function attack is clearly a violation of privacy; for instance, take the function $f_{\mathtt{ot}}$, where party P_1's input consists of a pair of strings ($x_1 = (\mathtt{z}_0, \mathtt{z}_1) \in \{0,1\}^{2\lambda}$), and party P_n's input consists of a bit ($x_n = c \in \{0,1\}$). All parties receive $\mathtt{z}_c$ as output.

Assume, for the sake of contradiction, that the protocol Π is r-last message resilient, and realizes $f_{\mathtt{ot}}$ in two rounds without using private channels in the first round. Of course, in an all-honest execution, P_n learns $\mathtt{z}_c$. By r-last message resiliency (for $r \geq 1$), P_n can compute $\mathtt{z}_c$ *even without P_1's second-round message.*

Now, an adversary observing the network can mount a residual attack by eavesdropping all of P_1's first-round messages, and running the rest of the protocol in her head. More concretely, she can compute first- and second-round messages on behalf of all the other participants $P_2, \ldots, P_n$ (using arbitrary inputs $x'_2, \ldots, x'_n$ of her choice), and run the rest of the protocol, ending up with the output $\mathtt{z}_0$ if $x'_n = 0$ and $\mathtt{z}_1$ if $x'_n = 1$. The absence of a second-round message from P_1 should not affect the output by r-last message resiliency.

The above proof breaks down in the presence of a PKI or private peer-to-peer channels in the first round. This is because in such a case, the adversary would not be able to emulate the messages on behalf of parties $P_2, \ldots, P_n$ with respect to inputs of her choice. If a PKI is available, their messages may need to depend on their secret keys, which the adversary does not know; if private channels are available in the first round, their messages may depend on private information that they receive from P_1 in that round.

Corollary 1 (BC-P2P, UA, CRS, $t \geq 1$). *There exists a function f such that no n-party two-round protocol Π can compute f with unanimous abort against $t \geq 1$ corruptions while making use of broadcast and public peer-to-peer channels in the first round and (private) peer-to-peer channels in the second round, as long as $n - t \geq 2$ (that is, as long as there are at least two honest parties).*

This follows from Theorem 6 and Theorem 8.

Corollary 2 (BC-BC, FAIR, CRS, $t \geq 1$). *There exists a function f such that no n-party two-round protocol Π can compute f with fairness against $t \geq 1$ corruptions while making use of broadcast and public peer-to-peer channels in the first round and both broadcast and (private) peer-to-peer channels in the second round.*

This follows from Theorem 7 and Theorem 8.

References

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.

BHR12a. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 134–153. Springer, Heidelberg, December 2012.

BHR12b. 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.

BMMR21. Saikrishna Badrinarayanan, Peihan Miao, Pratyay Mukherjee, and Divya Ravi. On the round complexity of fully secure solitary mpc with honest majority. Cryptology ePrint Archive, Report 2021/241, 2021. `https://eprint.iacr.org/2021/241`.

CCD+20. Megan Chen, Ran Cohen, Jack Doerner, Yashvanth Kondi, Eysa Lee, Schuyler Rosefield, and abhi shelat. Multiparty generation of an RSA modulus. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 64–93. Springer, Heidelberg, August 2020.

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.

CL14. Ran Cohen and Yehuda Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 466–485. Springer, Heidelberg, December 2014.

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.

DDO+01. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Heidelberg, August 2001.

DMR+21. Ivan Damgård, Bernardo Magri, Divya Ravi, Luisa Siniscalchi, and Sophia Yakoubov. Broadcast-optimal two round MPC with an honest majority. In *Crypto*, LNCS, pages 155–184. Springer, Heidelberg, 2021.

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.

GJPR21. Aarushi Goel, Abhishek Jain, Manoj Prabhakaran, and Rajeev Raghunath. On communication models and best-achievable security in two-round MPC. *TCC*, page 690, 2021.

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.

GP16. Chaya Ganesh and Arpita Patra. Broadcast extensions with optimal communication and round complexity. In George Giakkoupis, editor, *35th ACM PODC*, pages 371–380. ACM, July 2016.

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.

HR14. Martin Hirt and Pavel Raykov. Multi-valued byzantine broadcast: The $t < n$ case. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 448–465. Springer, Heidelberg, December 2014.

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.

LSP82. Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

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.

Sha79. Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

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

A Building Blocks

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

A.1 Symmetric Key Encryption

Definition 5 (Symmetric-Key Encryption (SKE)). *A symmetric-key encryption (SKE) scheme is a tuple of efficient algorithms* $\mathsf{SKE} = (\mathtt{keygen}, \mathtt{enc}, \mathtt{dec})$ *defined as follows.*

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

We require the following properties of a symmetric encryption scheme:

Correctness. We say that SKE *satisfies* correctness *if for all* $\mathtt{sk} \leftarrow \mathtt{keygen}(1^\lambda)$*,*

$$\Pr[\mathtt{dec}(\mathtt{sk}, \mathtt{enc}(\mathtt{sk}, \mathsf{msg})) = \mathsf{msg}] \geq 1 - negl(\lambda)$$

(where the randomness is taken over the internal coin tosses of algorithm $\mathtt{enc}$*).*

Semantic Security. We say that SKE *satisfies* semantic security *if for all PPT adversaries* $\mathcal{A}$*, for* $(\mathsf{msg}_0, \mathsf{msg}_1) \leftarrow \mathcal{A}(1^\lambda)$*, if* $|\mathsf{msg}_0| = |\mathsf{msg}_1|$,

$$\Pr\left[\mathcal{A}(c) = b : \begin{array}{c} \mathtt{sk} \leftarrow \mathtt{keygen}(1^\lambda); b \leftarrow \{0,1\}; \\ c \leftarrow \mathtt{enc}(\mathtt{sk}, \mathsf{msg}_b); \end{array}\right] \leq \frac{1}{2} + negl(\lambda)$$

(where the randomness is taken over the internal coin tosses of $\mathcal{A}$*,* $\mathtt{keygen}$ *and* $\mathtt{enc}$*).*

Instantiation. For our constructions, we use SKE with deterministic encryption (such as one-time pad encryption) which satisfies perfect correctness.

A.2 Public Key Encryption

Definition 6 (Public-Key Encryption (PKE)). *A public-key encryption (PKE) scheme is a tuple of efficient algorithms* $\mathsf{PKE} = (\mathtt{keygen}, \mathtt{enc}, \mathtt{dec})$ *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* $(\texttt{pk}, \texttt{sk})$*.*
$\texttt{enc}(\texttt{pk}, \mathsf{msg}; r) \to c$**:** *The probabilistic algorithm* enc *takes as input the public key* pk*, a message* $\mathsf{msg} \in \mathcal{M}$*, and implicit randomness* $\rho \in \mathcal{R}$*, and outputs a ciphertext* $c = \texttt{enc}(\texttt{pk}, \mathsf{msg}; r)$*. The set of all ciphertexts is denoted by* $\mathcal{C}$*.*
$\texttt{dec}(\texttt{sk}, c) \to \mathsf{msg}$**:** *The deterministic algorithm* dec *takes as input the secret key* sk *and a ciphertext* $c \in \mathcal{C}$ *and outputs* $\mathsf{msg} = \texttt{dec}(\texttt{sk}, c)$ *which is either equal to some message* $\mathsf{msg} \in \mathcal{M}$ *or to an error symbol* $\perp$*.*

We require the following properties of a PKE scheme:

Correctness. We say that PKE *satisfies* correctness *if for all* $(\texttt{pk}, \texttt{sk}) \leftarrow \texttt{keygen}(1^\lambda)$*,*

$$\Pr[\texttt{dec}(\texttt{sk}, \texttt{enc}(\texttt{pk}, \mathsf{msg})) = \mathsf{msg}] = 1$$

(where the randomness is taken over the internal coin tosses of algorithm enc*).*

CPA Security. We say that PKE *satisfies* semantic security *if for all PPT adversaries* $\mathcal{A}$*, for* $(\mathsf{msg}_0, \mathsf{msg}_1) \leftarrow \mathcal{A}(1^\lambda)$*, if* $|\mathsf{msg}_0| = |\mathsf{msg}_1|$*,*

$$\Pr\left[\mathcal{A}(\texttt{pk}, c) = b : \begin{array}{c} (\texttt{pk}, \texttt{sk}) \leftarrow \texttt{keygen}(1^\lambda); b \leftarrow \{0,1\}; \\ c \leftarrow \texttt{enc}(\texttt{pk}, \mathsf{msg}_b) \end{array}\right] \leq \frac{1}{2} + negl(\lambda)$$

(where the randomness is taken over the internal coin tosses of $\mathcal{A}$*,* keygen *and* enc*).*

A.3 Non-Interactive Zero-Knowledge Arguments of Knowledge

We take this definition from Groth and Maller [GM17].

Definition 7 (Non-Interactive Zero-Knowledge Arguments of Knowledge (NIZKAoK)). *A non-interactive zero-knowledge argument of knowledge (*NIZK*) scheme is a tuple of efficient algorithms* NIZK = (setupZK, prove, verify, simP) *defined as follows.*

$\texttt{setupZK}(1^\lambda, \mathcal{R}) \to (crs, td)$**:** *The algorithm* setupZK *takes as input the security parameter* $\lambda \in \mathbb{N}$*, and outputs the global common reference string* crs *and the trapdoor* td *for the* NIZK *system.*
$\texttt{prove}(crs, \phi, w) \to \pi$**:** *The algorithm* prove *takes as input the common reference string* crs *for a relation* $\mathcal{R}$*, a statement* ϕ *and a witness* w*, and outputs a proof* π *that* $(\phi, w) \in \mathcal{R}$*.*
$\texttt{verify}(crs, \phi, \pi) \to \texttt{accept/reject}$**:** *The algorithm* verify *takes as input the common reference string* crs *for a relation* $\mathcal{R}$*, a statement* ϕ *and a proof* π*, and verifies whether* π *proves the existence of a witness* w *such that* $(\phi, w) \in \mathcal{R}$*.*
$\texttt{simP}(crs, td, \phi) \to \pi$**:** *The algorithm* simP *takes as input the common reference string* crs *for a relation* $\mathcal{R}$*, the trapdoor* td *and a statement* ϕ*, and outputs a simulated proof of the existence of a witness* w *such that* $(\phi, w) \in \mathcal{R}$*.*

We require the following properties of a NIZK *scheme:*

Correctness. We say that `NIZK` *satisfies* correctness *if for any* $(\phi, w) \in \mathcal{R}$*, we have that*

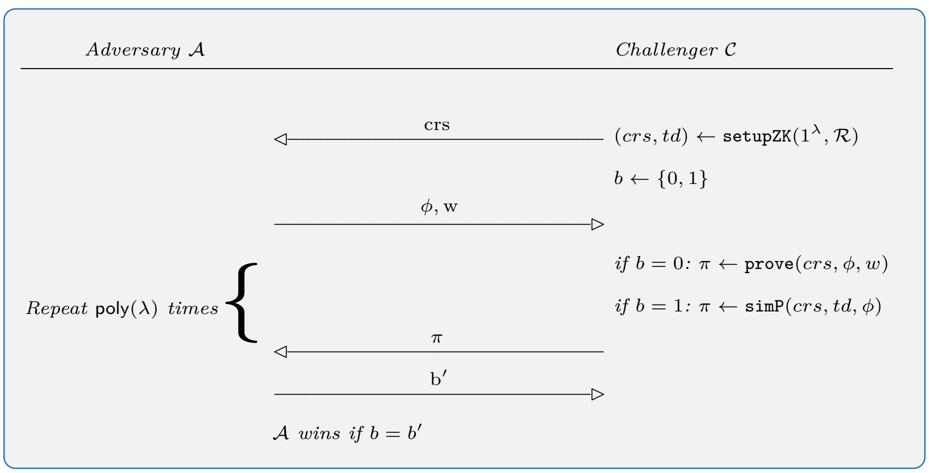
$$\Pr\left[\texttt{verify}(\phi,\pi)=1 \;\middle|\; \begin{array}{l}(crs,td)\leftarrow \texttt{setupZK}(1^\lambda,\mathcal{R})\\ \pi \leftarrow \texttt{prove}(\phi,w)\end{array}\right] \geq 1 - negl(\lambda)$$

(where the randomness is taken over the internal coin tosses of `setupZK`*,* `prove` *and* `verify`*).*

Zero Knowledge. We say that `NIZK` *satisfies* zero-knowledge *if for all PPT adversaries* $\mathcal{A}$*,*

$$\Pr[\mathcal{A}\ wins] \leq \frac{1}{2} + negl(\lambda)$$

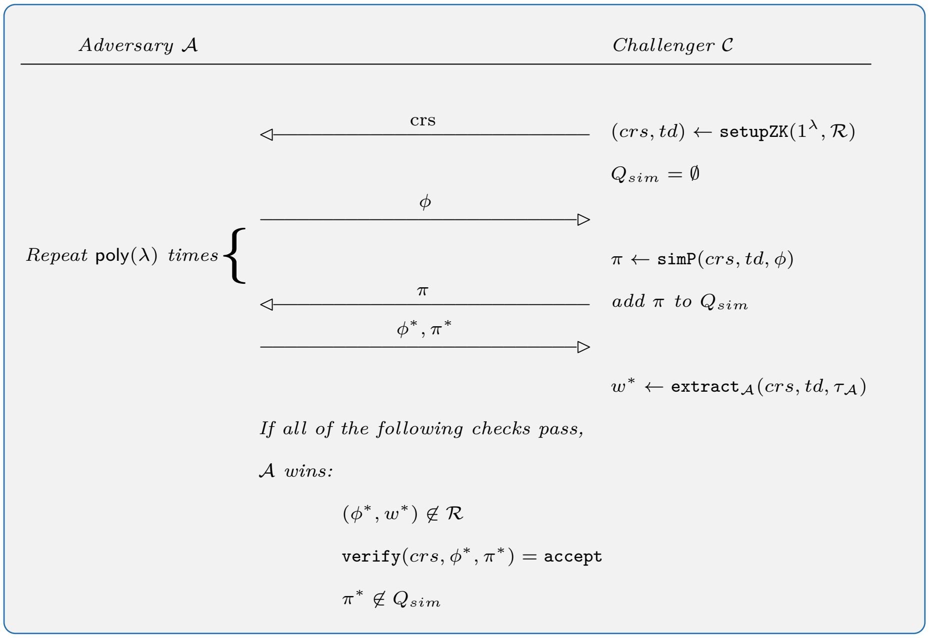
in the following experiment:



Simulation Extractability. We say that `NIZK` *satisfies* simulation extractability *if for all PPT adversaries* $\mathcal{A}$ *there exists a PPT extraction algorithm* $\texttt{extract}_{\mathcal{A}}$ *such that*

$$\Pr[\mathcal{A}\ wins] \leq negl(\lambda)$$

in the following experiment:



Instantiation. Simulation extractable NIZK could be instantiated using, for instance, technique from [DDO⁺01].

A.4 Garbling Scheme

A garbling scheme, introduced by Yao [Yao82] and formalized by Bellare *et al.* [BHR12b], enables a party to "encrypt" or "garble" a circuit in such a way that it can be evaluated on inputs — given tokens or "labels" corresponding to those inputs — without revealing what the inputs are.

Definition 8 (Garbling Scheme). *A projective garbling scheme is a tuple of efficient algorithms* $\texttt{GC} = (\texttt{garble}, \texttt{eval})$ *defined as follows.*

$\texttt{garble}(1^\lambda, \texttt{C}) \to (\texttt{GC}, \mathbf{K})$**:** *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 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:

Perfect Correctness. We say GC *satisfies* perfect correctness *if for any boolean circuit* $\mathtt{C} : \{0,1\}^\ell \to \{0,1\}^m$ *and* $x = (x_1, \ldots, x_\ell)$ *it holds that*

$$\Pr[\mathtt{eval}(\mathtt{GC}, \mathbf{K}[x]) = \mathtt{C}(x)] = 1,$$

where $(\mathtt{GC}, \mathbf{K}) \leftarrow \mathtt{garble}(1^\lambda, \mathtt{C})$ *with* $\mathbf{K} = (K_1^0, K_1^1, \ldots, K_\ell^0, K_\ell^1)$*, 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. When garbled circuits are used in such a way that decoding information is used separately, obliviousness *requires that a garbled circuit together with a set of labels reveals nothing about the input the labels correspond to, and* privacy *requires that the additional knowledge of the decoding information reveals only the appropriate output. In our work, we do not consider decoding information separately (but rather, consider it to be included in the garbled circuit), so we do not need obliviousness.*

Privacy. Informally, privacy requires that a garbled circuit together with a set of labels reveal nothing about the input the labels correspond to (beyond the appropriate output).

More formally, we say that GC *satisfies* privacy *if there exists a simulator* simGC *such that for every PPT adversary* $\mathcal{A}$*, it holds that*

$$\Pr[\mathcal{A} \text{ wins}] \leq \frac{1}{2} + negl(\lambda)$$

in the following experiment:

Adversary $\mathcal{A}$		*Challenger* $\mathcal{C}$
	$\mathtt{C} : \{0,1\}^\ell \to \{0,1\}^m$ →	
	$x = (x_1, \ldots, x_\ell) \in \{0,1\}^\ell$ →	$b \leftarrow \{0,1\}$
		if $b = 0$*:*
		$(\mathtt{GC}, (K_1^0, K_1^1, \ldots, K_\ell^0, K_\ell^1)) \leftarrow \mathtt{garble}(1^\lambda, \mathtt{C})$
		$K_i = K_i^{x_i}$ *for* $i \in [\ell]$
		if $b = 1$*:*
		$(\mathtt{GC}, K_1, \ldots, K_\ell) \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}, \mathtt{C}(x))$
	← $\mathtt{GC}, K_1, \ldots, K_\ell$	
	b' →	
	$\mathcal{A}$ *wins if* $b = b'$	

Remark. *It is possible to use an alternate variant of the simulator* simGC *that takes as input a set of labels* $(K_1, \ldots, K_\ell)$ *and returns a garbled circuit* GC *compatible with these labels. The simulator of Yao's [Yao82] garbling scheme can be made to work easily as mentioned above.*

Adaptive Privacy. Informally, this property requires that privacy is maintained against an adversary who first obtains the garbled circuit and then selects the input. More formally, we say that GC *satisfies* adaptive privacy *if there exists a simulator* simGC *such that for every PPT adversary* $\mathcal{A}$*, it holds that*

$$\Pr[\mathcal{A} \text{ wins}] \leq \frac{1}{2} + negl(\lambda)$$

in the following experiment:

Adversary $\mathcal{A}$		*Challenger* $\mathcal{C}$
	$\mathtt{C} : \{0,1\}^\ell \to \{0,1\}^m$ →	$b \leftarrow \{0,1\}$
		if $b = 0$:
		$(\mathtt{GC}, (K_1^0, K_1^1, \ldots, K_\ell^0, K_\ell^1)) \leftarrow \mathtt{garble}(1^\lambda, \mathtt{C})$
		if $b = 1$:
		$\mathtt{GC} \leftarrow \mathtt{simGC}(1^\lambda, \phi(\mathtt{C}), \text{"}ckt\text{"})$
		where $\phi(\mathtt{C})$ *denotes the topology of* C [a]
	← GC	
	$x = (x_1, \ldots, x_\ell) \in \{0,1\}^\ell$ →	*if* $b = 0$:
		$K_i = K_i^{x_i}$ *for* $i \in [\ell]$
		if $b = 1$:
		$((K_1, \ldots, K_\ell) \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}(x), \text{"}input\text{"})$
	← $K_1, \ldots, K_\ell$	
	b' →	
$\mathcal{A}$ *wins if* $b = b'$		

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

Instantiation. For our constructions, adaptive garbled circuits can be obtained using one-time pads with Yao's garbled circuits (as shown by Bellare *et al.* [BHR12a]).

A.5 Threshold 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 $t + 1$ shares need to be used.

Definition 9 (Secret Sharing). *A t-out-of-n secret sharing scheme is a tuple of efficient algorithms* (share, reconstruct) *defined as follows.*

$\mathtt{share}(x) \to (s_1, \ldots, s_n)$**:** *The randomized algorithm* `share` *takes as input a secret x and output a set of n shares.*

$\mathtt{reconstruct}(\{s_i\}_{i \in S \subseteq [n], |S| > t}) \to x$**:** *The reconstruct algorithm* `reconstruct` *takes as input a vector of at least $t+1$ shares and outputs the secret x.*

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

Perfect Correctness. The perfect correctness property requires that the shares of a secret x should always reconstruct to x. More formally, a secret sharing scheme is perfectly correct *if for any secret x, for any subset $S \subseteq [n], |S| > t$,*

$$\Pr\left[x = x' : \begin{array}{c} (s_1, \ldots, s_n) \leftarrow \mathtt{share}(x) \\ x' \leftarrow \mathtt{reconstruct}(\{s_i\}_{i \in S}) \end{array} \right] = 1,$$

where the probability is taken over the random coins of `share`*. Moreover, if a negligible error probability is allowed, we simply say that the scheme is correct.*

Privacy. The privacy property requires that any combination of up to t shares should leak no information about the secret x. More formally, we say that a secret sharing scheme is private *if for all (unbounded) adversaries $\mathcal{A}$, for any set $\mathbb{A} \subseteq \{1, \ldots, n\}$, $|\mathbb{A}| \leq t$ and any two secrets x_0, x_1 (such that $|x_0| = |x_1|$),*

$$\Pr\left[\mathcal{A}(\mathbf{s}) = 1 : \left| \begin{array}{c} \{s_i\}_{i \in [n]} = \mathtt{share}(x_0); \\ \mathbf{s} = \{s_i\}_{i \in \mathbb{A}} \end{array} \right. \right] \equiv \Pr\left[\mathcal{A}(\mathbf{s}) = 1 : \left| \begin{array}{c} \{s_i\}_{i \in [n]} = \mathtt{share}(x_1); \\ \mathbf{s} = \{s_i\}_{i \in \mathbb{A}} \end{array} \right. \right].$$

Share Simulatability. Additionally, we require an efficient simulator for the generated shares. More formally, we say that a secret sharing scheme is share simulatable *if there exists a PPT simulator* `simshare` *such that for every PPT adversary $\mathcal{A}$, for any set $\mathbb{A} \subseteq \{1, \ldots, n\}$, $|\mathbb{A}| \leq t$ (and $\mathbb{H} = \{1, \ldots, n\} \setminus \mathbb{A}$), and any two secrets x_0, x_1, for $(s_0, \ldots, s_n) \leftarrow \mathtt{share}(x_0)$, $(s'_1, \ldots, s'_n) \leftarrow \mathtt{share}(x_1)$ and $\{s''_i\}_{i \in \mathbb{H}} \leftarrow \mathtt{simshare}(\{s_i\}_{i \in \mathbb{A}}, x_0)$,*

$$|\Pr[\mathcal{A}(\{s_i\}_{i \in \mathbb{A}}, \{s_i\}_{i \in \mathbb{H}}) = 1] - \Pr[\mathcal{A}(\{s_i\}_{i \in \mathbb{A}}, \{s''_i\}_{i \in \mathbb{H}}) = 1]| \leq negl(\lambda).$$

Instantiation. In our constructions, we use the Shamir's threshold secret sharing scheme [Sha79], and refer to its algorithms as ($\mathtt{Shamir}.s, \mathtt{Shamir}.\mathtt{reconstruct}$).

B Formal Definitions of One-or-Nothing Secret Sharing with Intermediaries

In this appendix, we give the formal definitions of one-or-nothing secret sharing with intermediaries.

Definition 10 **(**`1or0wi`**: Privacy).**

A one-or-nothing secret sharing with intermediaries scheme is private *if for any security parameter* $\lambda \in \mathbb{N}$*, for every PPT adversary* $\mathcal{A}$*, it holds that* $\Pr[\mathcal{A}\ wins] \leq \frac{1}{2} + negl(\lambda)$ *in the following experiment:*

$\mathcal{A}$		Challenger $\mathcal{C}$
		$b \leftarrow \{0,1\}$
	$\mathbb{A} \subset \{1,\ldots,n-1\}$ $(s.t.\ \lvert\mathbb{A}\rvert \leq t)$ $\longrightarrow$	$\mathbb{H} := \{D,1,\ldots,n-1\} \backslash \mathbb{A}$
		$crs \leftarrow \texttt{setup}(1^\lambda)$
		$(\texttt{sk}_i, \texttt{pk}_i) \leftarrow \texttt{keygen}(crs)$ *for* $i \in \mathbb{H}$
	$\longleftarrow$ $crs, \{\texttt{pk}_i\}_{i\in\mathbb{H}}$	
	$v, \mathbf{z}_0^{(v)}, \mathbf{z}_1^{(v)} (s.t.\ \lvert\mathbf{z}_0^{(v)}\rvert = \lvert\mathbf{z}_1^{(v)}\rvert)$ $\longrightarrow$	$\mathbf{z}^{(v)} := \mathbf{z}_b^{(v)}$
	$\{\mathbf{z}^{(v')}\}_{v' \in \{1,\ldots,l\}\backslash\{v\}}$ $\longrightarrow$	
	$\{\texttt{pk}_{j\to i}\}_{j\in\mathbb{A}, i\in\mathbb{H}}, \{v_i\}_{i\in\mathbb{H}}$ $\longrightarrow$	
		Let $\texttt{pk}_{i\to j} = \texttt{pk}_i$ *for* $i,j \in \mathbb{H}$
		$s \leftarrow \texttt{share}(crs, \texttt{pk}_{1\to D}, \ldots, \texttt{pk}_{n\to D}, \mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(l)})$
		For $i \in \mathbb{H}$*:*
		$\overline{s}_i \leftarrow \texttt{vote}(crs, \texttt{sk}_i, \texttt{pk}_{1\to i}, \ldots, \texttt{pk}_{n\to i}, v_i)$
	$\longleftarrow$ $s, \{\overline{s}_i\}_{i\in\mathbb{H}}$	
	b' $\longrightarrow$	
Let S *denote set of indices* $i \in \mathbb{H}$ *such that* $v_i = v$*.* $\mathcal{A}$ *wins if* $b' = b$ *and* $\lvert S\rvert < n - 2t$*.*		

Definition 11 **(**`1or0wi`**: Identifiability).**

A one-or-nothing secret sharing with intermediaries scheme is identifiable *if for any security parameter* $\lambda \in \mathbb{N}$*, for every PPT adversary* $\mathcal{A}$*, it holds that* $\Pr[\mathcal{A}\ wins]$ *is negligible in the following experiment:*

$\mathcal{A}$		*Challenger* $\mathcal{C}$
$\mathbb{A} \subset \{D, 1, \ldots, n-1\}$ (*s.t.* $\lvert\mathbb{A}\rvert \leq t$)	$\longrightarrow$	$\mathbb{H} := \{D, 1, \ldots, n-1\} \backslash \mathbb{A}$
		$crs \leftarrow \texttt{setup}(1^\lambda)$
		$(\texttt{sk}_i, \texttt{pk}_i) \leftarrow \texttt{keygen}(crs)$ *for* $i \in \mathbb{H}$
$crs, \{\texttt{pk}_i\}_{i \in \mathbb{H}}$	$\longleftarrow$	
$\{\texttt{z}^{(v)}\}_{v \in \{1,\ldots,l\}}$ *if* $D \in \mathbb{H}$	$\longrightarrow$	
$\{\texttt{pk}_{j \to i}\}_{j \in \mathbb{A}, i \in \mathbb{H}}, v, \{v_i\}_{i \in \mathbb{H}}$	$\longrightarrow$	
		Let $\texttt{pk}_{i \to j} = \texttt{pk}_i$ *for* $i, j \in \mathbb{H}$
		If $D \in \mathbb{H}$*:*
		$s \leftarrow \texttt{share}(crs, \texttt{pk}_{1 \to D}, \ldots, \texttt{pk}_{n \to D}, \texttt{z}^{(1)}, \ldots, \texttt{z}^{(l)})$
		For $i \in \mathbb{H}$*:*
		$\overline{s}_i \leftarrow \texttt{vote}(crs, \texttt{sk}_i, \texttt{pk}_{1 \to i}, \ldots, \texttt{pk}_{n \to i}, v_i)$
s *if* $D \in \mathbb{H}, \{\overline{s}_i\}_{i \in \mathbb{H}}$	$\longleftarrow$	
$\{(\texttt{pk}_i, v_i, \overline{s}_i)\}_{i \in \mathbb{A}}$	$\longrightarrow$	
		$z \leftarrow \texttt{reconstruct}\Big(crs, s, (\texttt{pk}_1, v_1, \overline{s}_1), \ldots, (\texttt{pk}_n, v_n, \overline{s}_n)\Big)$
z	$\longleftarrow$	
$\mathcal{A}$ *wins if there exists a set* S *with* $\lvert S\rvert \geq n - t$ *such that* $v_i = v$ *for* $i \in S$ *and* z *is neither (a)* $\texttt{z}^{(v)}$ *nor (b)* $\bot_i$*, where* $i \in \mathbb{A}$		

C Proof of Theorem 1

We prove privacy and identifiability (which implies correctness) below.

Privacy. Let $\mathbb{H}$ denote the set of indices of honest parties and $S \subset \mathbb{H}$ (where $\lvert S\rvert \leq n - 2t - 1$) denote the set of indices of honest parties that produce ballot for v. We show that the adversary learns nothing about $s_i^{(v)}$ for any $i \in \mathbb{H} \setminus S$. This would suffice to show that the adversary learns nothing about $\texttt{z}^{(v)}$. This is because the adversary would have access to at most $t + \lvert S\rvert \leq t + (n - 2t - 1) = n - t - 1$ shares of $\texttt{z}^{(v)}$ (which is shared using threshold $(n - t - 1)$). Privacy of threshold sharing dictates that the adversary learns nothing about $\texttt{z}^{(v)}$.

Consider an honest party $i \in \mathbb{H} \setminus S$. Suppose party i votes for $v_i \neq v$. Firstly, we argue that the adversary learns nothing about $\texttt{sk}_{i \to j}^{(v)}$ for any $j \in \mathbb{H}$

as follows : Based on the specifications of vote, honest P_i reveals nothing about $\mathtt{sk}_{i\to i}^{(v)}$. It now follows from the CPA security of the PKE that the adversary learns nothing about $\mathtt{sk}_{i\to j}^{(v)}$ from $\mathtt{tk}_{i\to j}^{(v)}$ (contained in $\bar{s}_j$). Further, the zero-knowledge property of NIZK generated by honest P_j ensures that the adversary learns nothing about $\mathtt{sk}_{i\to j}^{(v)}$ from $\{\pi_{i\to j}^{(v)}\}$ (contained in $\mathtt{pk}_j$) and $\pi_{\mathtt{vote},j}$. We can now infer from CPA security of the PKE and zero-knowledge property of the NIZK generated by the honest dealer, that the adversary learns nothing about $\{s_{i\to j}^{(v)}\}_{j\in\mathbb{H}}$ from $\{c_{i\to j}^{(v)}\}_{j\in\mathbb{H}}$ and $\pi_{\mathtt{share}}$ respectively.

We can thus conclude that the adversary has access to at most t shares of $s_i^{(v)}$ which is shared using threshold $(n-2t-1) \geq t$. It now follows from privacy of threshold sharing that the adversary learns nothing about $s_i^{(v)}$; completing the proof.

Identifiability. Suppose $(n-t)$ parties, say constituting the set $S_{\mathtt{vote}}$, produce ballot using the same v. Let $\mathbb{H}$ denote the set of indices corresponding to honest parties.

First, we argue that $\bot_i$, where $i \in \mathbb{H}$ is output with negligible probability. We observe that this can happen only when one of the following holds **(a)** NIZK proof $\pi_{\mathtt{share}}$ (if i is the dealer) or $\pi_{\mathtt{vote},i}$ or $\pi_{j\to i}^{(v)}$ (for any $j \in [n]$) does not verify *or* **(b)** $|S_i'| < n-t$ *or* **(c)** i is the dealer and $|S_D| < n-t$. First, it directly follows from correctness of NIZK that **(a)** cannot occur with respect to an honest P_i, except with negligible probability. Next, we note that $\mathbb{H} \subseteq S_i'$. This is because each pair of honest parties will be in agreement with respect to each other's public keys. Thus, $|S_i'| \geq n-t$, implying that **(b)** cannot hold. Similarly, an honest dealer would also be in agreement with all the honest parties with respect to their public keys. Thus, $|S_D| \geq n-t$, implying that **(c)** cannot hold. This completes the argument that $\bot_i$, where $i \in \mathbb{H}$ is output with negligible probability. From the above, we can conclude that when $\bot_i$ is output, $i \notin \mathbb{H}$ with overwhelming probability. Therefore, to complete the proof, it suffices to show that when no cheater is identified, $\mathtt{z}^{(v)}$ is reconstructed.

Suppose no cheating party is identified. Since $|S_D| \geq n-t > 2t$ must hold, S_D constitutes honest parties who must have verified $\pi_{\mathtt{share}}$ by a potentially corrupt dealer. It follows from simulation-soundness of NIZK that the encryptions $\{c_{k\to j}^{(v)}\}_{k,j\in[n]}$ must have indeed been computed correctly. Next, consider $k \in S_{\mathtt{vote}}$ and $j \in S_k$ (where $S_k = S_k' \cap S_D$).

Recall that P_k and P_j are in mutual agreement with respect to their public keys and $(\pi_{\mathtt{vote},k}, \pi_{k\to k}^{(v)})$ sent by P_k and $(\pi_{\mathtt{vote},j}, \pi_{k\to j}^{(v)})$ sent by P_j verified successfully (otherwise a party must have been identified as cheater). Simulation soundness of NIZK ensures that the public key $\mathtt{pk}_{k\to k}^{(v)}$ used by P_j to broadcast ciphertext $\mathtt{tk}_{k\to j}^{(v)}$ (contained in $\bar{s}_j$) corresponds to the secret key $\mathtt{sk}_{k\to k}^{(v)}$ (contained in $\bar{s}_k$) broadcast by P_k. It now follows from the correctness of the encryption scheme that $\mathtt{sk}_{k\to j}^{(v)}$ is obtained upon decrypting $\mathtt{tk}_{k\to j}^{(v)}$ with overwhelming probability. Further, since $j \in S_D$, it follows from simulation-soundness of the NIZK $\pi_{\mathtt{share}}$ and $\pi_{\mathtt{vote},j}$ that the public key $\mathtt{pk}_{k\to j}^{(v)}$ (used by dealer for the ciphertext

$c^{(v)}_{k\to j}$) corresponds to the secret key $\texttt{sk}^{(v)}_{k\to j}$ (used by P_j during $\texttt{vote}$). It thus follows $s^{(v)}_{k\to j}$ is obtained upon decrypting $c^{(v)}_{k\to j}$ with overwhelming probability.

Next, since $|S_k| \geq n-2t$ and $s^{(v)}_k$ is shared using threshold $(n-2t-1)$, it follows from correctness of Shamir's threshold sharing that $s^{(v)}_k \neq \bot$ is reconstructed successfully for all $k \in S_{\texttt{vote}}$. Lastly, since $|S_{\texttt{vote}}| \geq n-t$, $\texttt{z}^{(v)} \neq \bot$ which is shared using threshold $(n-t-1)$ is also reconstructed successfully (due to correctness of shamir's threshold sharing). This completes the proof.

D Proof of Theorem 2

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

We assume that an adversary is deterministic and that the output of the adversary consists of her entire view during the protocol, i.e., the auxiliary information, the input, and the CRS setup (which includes the CRS setup if required by the underlying protocol $\Pi_{\texttt{bc}}$) of all corrupt parties, and the messages received by honest parties during the protocol.

We start by giving the description of a receiver specific adversary ($\mathcal{A}_q$, where $q \in \mathbb{H}$), which is an adversary against $\Pi_{\texttt{bc}}$. Then we give the description of our simulator $\mathcal{S}$ which will make use of the simulator $\mathcal{S}_q$ against the adversary $\mathcal{A}_q$, for the minimal index $q \in \mathbb{H}$.

Figure D.1: The Receiver Specific Adversary $\mathcal{A}_q$

Setup.

- $\mathcal{A}_q$ runs setup for one-or-nothing secret sharing with intermediaries: $crs \leftarrow \texttt{setup}(1^\lambda)$. The setup of the underlying protocol $\Pi_{\texttt{bc}}$ is also run.

First Round. For each $i \in \mathbb{H}$, upon receiving the first-broadcast- round message msg^1_i from an honest party P_i in $\Pi_{\texttt{bc}}$; $\mathcal{A}_q$ computes the following steps:

1. Let $(\texttt{sk}_i, \texttt{pk}_i) \leftarrow \texttt{keygen}(1^\lambda)$, where $\texttt{pk}_i = \{\texttt{pk}^{(1)}_i = (\texttt{pk}^{(1,1)}_i, \ldots, \texttt{pk}^{(1,\mathsf{L})}_i), \ldots, \texttt{pk}^{(n)}_i = (\texttt{pk}^{(n,1)}_i, \ldots, \texttt{pk}^{(n,\mathsf{L})}_i)\}$ is a vector of $n\mathsf{L}$ public keys with the corresponding vector of secret keys $\texttt{sk}_i = \{\texttt{sk}^{(1)}_i = (\texttt{sk}^{(1,1)}_i, \ldots, \texttt{sk}^{(1,\mathsf{L})}_i), \ldots, \texttt{sk}^{(n)}_i = (\texttt{sk}^{(n,1)}_i, \ldots, \texttt{sk}^{(n,\mathsf{L})}_i)\}$
2. Send $(\texttt{pk}_i, \mathsf{msg}^1_i)$ to $\mathcal{A}$ in $\Pi^{\mathsf{id\text{-}abort}}_{\mathsf{p2pbc}}$.
3. For each party P_j s.t. $j \in \mathbb{A}$: Let $\{\mathsf{msg}^1_{j\to i}\}_{i\in\mathbb{H}}$ be the messages received by $\mathcal{A}_q$ (which is acting on behalf of honest parties in $\Pi^{\mathsf{id\text{-}abort}}_{\mathsf{p2pbc}}$) in the first round. $\mathcal{A}_q$ scans the messages $\{\mathsf{msg}^1_{j\to i}\}_{i\in\mathbb{H}}$ bit by bit and $\forall\beta \in \ell$ checks what is the β-th bits b^β_j that is consistent among $(\frac{1}{3}n+1)$ of the messages $\{\mathsf{msg}^1_{j\to i}\}_{i\in\mathbb{H}}$. If it does not exist such a bit (which implies that less than $\frac{1}{3}n$ of the messages were consistent) $\mathcal{A}_q$ sets $\mathsf{msg}^1_j = \mathsf{msg}_{j\to q}$, otherwise $\mathcal{A}_q$ sets $\mathsf{msg}^1_j = b^1_j \ldots, b^\ell_j$.

Otherwise she sets $\mathsf{msg}_j^1 = \mathsf{msg}_{j \to q}$. $\mathcal{A}_q$ broadcasts msg_j^1 in $\Pi_{\mathtt{bc}}$.

Second Round. For each $i \in \mathbb{H}$, upon receiving the second-broadcast-round message msg_i^2 from an honest party P_i in $\Pi_{\mathtt{bc}}$; $\mathcal{A}_q$ computes the following steps:

We specify multiple broadcast messages separately for clarity; however, they are all sent simultaneously as a single round of communication.

1. Let $\mathtt{pk}_{j\to i} = \{\mathtt{pk}_{j\to i}^{(1)}, \ldots, \mathtt{pk}_{j\to i}^{(n)}\}$ denote the $\mathtt{pk}_j$ received privately from P_j ($j \in [n]$), where $\mathtt{pk}_{j\to i}^{(k)} = (\mathtt{pk}_{j\to i}^{(k,1)}, \ldots, \mathtt{pk}_{j\to i}^{(k,\mathsf{L})})$ for $k \in [n]$.
2. Let $(\nu_{i,z+1}, \ldots, \nu_{i,\mathsf{L}})$ denote the bits comprising $(\mathsf{msg}_{1\to i}^1, \ldots, \mathsf{msg}_{n\to i}^1)$, where $\mathsf{msg}_{j\to i}^1$ refers to msg_j^1 received from P_j in Round 1.
3. For each $k \in \mathbb{A}$ and $l \in [z+1, \mathsf{L}]$: Compute and broadcast (in $\Pi_{\mathsf{p2pbc}}^{\mathsf{id\text{-}abort}}$ to $\mathcal{A}$) $(\overline{s}_{i,l}^{(k)}) \leftarrow \mathtt{vote}(crs, \mathtt{sk}_i^{(k,l)}, \mathtt{pk}_{1\to i}^{(k,l)}, \ldots, \mathtt{pk}_{n\to i}^{(k,l)}, \nu_{i,l})$
4. Let $\mathtt{C}_i$ be the circuit computing $\mathtt{snd\text{-}msg}_i$ run $(\mathtt{GC}_i, \{\tilde{K}_{i,l}\}_{l\in[\mathsf{L}]}) \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}_i, \mathsf{msg}_i^2)$.
5. For every $l \in [z+1, \ldots, \mathsf{L}]$, broadcast $s_{i,l} \leftarrow \mathtt{share}(crs, \mathtt{pk}_{1\to i}^{(i,l)}, \ldots, \mathtt{pk}_{n\to i}^{(i,l)}, K_{i,l}^{(0)}, K_{i,l}^{(1)})$, where $K_{i,l}^{\nu_{i,l}} = \tilde{K}_{i,l}$ and $K_{i,l}^{1-\nu_{i,l}}$ is chosen at random.
6. Broadcast $(\mathtt{GC}_i, \{\tilde{K}_{i,l}\}_{l\in[z]})$ in $\Pi_{\mathsf{p2pbc}}^{\mathsf{id\text{-}abort}}$ to $\mathcal{A}$.
7. Broadcast $(\mathsf{msg}_{1\to i}^1, \ldots, \mathsf{msg}_{n\to i}^1)$ in $\Pi_{\mathsf{p2pbc}}^{\mathsf{id\text{-}abort}}$to $\mathcal{A}$.

Output Computation. $\mathcal{S}$ on behalf of each party P_i, where $i \in \mathbb{H}$, does the following:

1. For $j \in [n]$: Check if $\{\mathsf{msg}_{k\to j}^1\}_{k\in[n]}$ broadcast by P_j is consistent with $\{\overline{s}_{j,l}^{(k)}\}_{k\in[n], l\in[z+1,\mathsf{L}]}$. Output $\mathtt{abort}_j$ if the check fails.
   Decrypt the shares:
2. For $k \in [n]$ (whose garbled circuit we will now consider):
   (a) For $l \in [z+1, \mathsf{L}]$, $K_{k,l} \leftarrow \mathtt{reconstruct}(crs, s_{k,l}, (\mathtt{pk}_1^{(k,l)}, v_{1,l}, \overline{s}_{1,l}^{(k)}), \ldots, (\mathtt{pk}_n^{(k,l)}, v_{n,l}, \overline{s}_{n,l}^{(k,l)})$.
   (b) Evaluate $\mathsf{msg}_k^2 \leftarrow \mathtt{eval}(\mathtt{GC}_k, (K_{k,1}, \ldots, K_{k,\mathsf{L}}))$.
3. Finally, $\mathcal{A}_q$ broadcasts the messages msg_j^2 for every corrupt P_j in $\Pi_{\mathtt{bc}}$, outputs whatever $\mathcal{A}$ outputs, and halts.

By the security of $\Pi_{\mathtt{bc}}$, for every $q \in \mathbb{H}$ there exists a simulator $\mathcal{S}_q$ for the adversarial strategy $\mathcal{A}_q$ 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 $\mathcal{S}_q$ starts by extracting corrupt parties' input values $\vec{x'_k} = \{x'_{i,k}\}_{i\in\mathbb{A}}$, and sending them to her trusted party. Upon receiving the output value y, the simulator $\mathcal{S}_q$ sends a message $\mathtt{abort}_j$/continue (for some $j \in \mathbb{A}$), and finally outputs the simulated view of the adversary, consisting of its input and the simulated messages of $\Pi_{\mathtt{bc}}$:

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

Our simulator $\mathcal{S}$ will make use of $\mathcal{S}_{\mathsf{RS}}$ that is the simulator $\mathcal{S}_q$ where q is the minimal index s.t. $q \in \mathbb{H}$.

Figure D.2: Simulator $\mathcal{S}$

The simulator $\mathcal{S}$ starts by invoking $\mathcal{S}_{\mathsf{RS}}$ and simulating for $\mathcal{S}_{\mathsf{RS}}$ the interaction with $\mathcal{A}_q$ making use of the adversary $\mathcal{A}$ as described above. $\mathcal{S}$ receives back (from $\mathcal{S}_{\mathsf{RS}}$) $\vec{x} = \{x_i\}_{i\in\mathbb{A}}$ or an $\mathtt{abort}_j$, for some $j \in \mathbb{A}$. $\mathcal{S}$ simulates the interaction between $\mathcal{S}_{\mathsf{RS}}$ and the ideal functionality, relying on the trusted third party that computes $\mathcal{F}$. Specifically, $\mathcal{S}$ forwards the messages (e.g. extracted inputs or abort messages) that she receives from $\mathcal{S}_{\mathsf{RS}}$ to the trusted third party, and if $\mathcal{S}_{\mathsf{RS}}$ did not abort she receives y in response. $\mathcal{S}$ forwards y to $\mathcal{S}_{\mathsf{RS}}$ which outputs the simulated view:

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

After invoking $\mathcal{S}_{\mathsf{RS}}$ (as described above) to get simulated honest party first round messages, the simulator $\mathcal{S}$ executes the following steps. Note that because the adversary $\mathcal{A}$ is deterministic, the executions above and below will always result in the same adversarial behavior (and transcript) up until the second round.

Setup.

- CRS setup for one-or-nothing secret sharing with intermediaries : $crs \leftarrow \mathtt{setup}(1^\lambda)$. The setup of the underlying protocol Π_{bc} is also run.

First Round. $\mathcal{S}$ on behalf of each party P_i, where $i \in \mathbb{H}$, does the following:

1. Let $(\mathtt{sk}_i, \mathtt{pk}_i) \leftarrow \mathtt{keygen}(1^\lambda)$, where $\mathtt{pk}_i = \{\mathtt{pk}_i^{(1)} = (\mathtt{pk}_i^{(1,1)}, \ldots, \mathtt{pk}_i^{(1,\mathsf{L})}), \ldots, \mathtt{pk}_i^{(n)} = (\mathtt{pk}_i^{(n,1)}, \ldots, \mathtt{pk}_i^{(n,\mathsf{L})})\}$ is a vector of $n\mathsf{L}$ public keys with the corresponding vector of secret keys $\mathtt{sk}_i = \{\mathtt{sk}_i^{(1)} = (\mathtt{sk}_i^{(1,1)}, \ldots, \mathtt{sk}_i^{(1,\mathsf{L})}), \ldots, \mathtt{sk}_i^{(n)} = (\mathtt{sk}_i^{(n,1)}, \ldots, \mathtt{sk}_i^{(n,\mathsf{L})})\}$.
2. Send $(\mathtt{pk}_i, \hat{\mathsf{msg}}^1_i)$ to P_j for $j \in \mathbb{A}$.

Second Round. Let $\{\mathsf{msg}^1_{j\to i}\}_{i\in\mathbb{H},j\in\mathbb{A}}$ be the messages received by $\mathcal{S}$ in the first round. For all $i \in \mathbb{A}$ $\mathcal{S}$ scans the messages $\{\mathsf{msg}^1_{j\to i}\}_{i\in\mathbb{H}}$ bit by bit and $\forall\beta \in \ell$ checks what is the β-th bits b^β_j that is consistent among $(\frac{1}{3}n + 1)$ of the messages $\{\mathsf{msg}^1_{j\to i}\}_{i\in\mathbb{H}}$. If it does not exist such a bit (which implies that less than $\frac{1}{3}n$ of the messages were consistent) $\mathcal{S}$ sets $\mathsf{flag} = 0$, otherwise $\mathcal{S}$ sets $\mathsf{msg}^1_{j\to i} = b^1_j, \ldots, b^\ell_j$ and $\mathsf{flag} = 1$.
$\mathcal{S}$ on behalf of each party P_i, where $i \in \mathbb{H}$, does the following:
We specify multiple broadcast messages separately for clarity; however, they are all sent simultaneously as a single round of communication.

1. Let $\mathtt{pk}_{j\to i} = \{\mathtt{pk}^{(1)}_{j\to i}, \ldots, \mathtt{pk}^{(n)}_{j\to i}\}$ denote the $\mathtt{pk}_j$ received privately from P_j ($j \in [n]$), where $\mathtt{pk}^{(k)}_{j\to i} = (\mathtt{pk}^{(k,1)}_{j\to i}, \ldots, \mathtt{pk}^{(k,\mathsf{L})}_{j\to i})$ for $k \in [n]$.

2. Let $(\nu_{i,z+1}, \ldots, \nu_{i,\mathsf{L}})$ denote the bits comprising $(\mathsf{msg}^1_{1\to i}, \ldots, \mathsf{msg}^1_{n\to i})$, where $\mathsf{msg}^1_{j\to i}$ refers to msg^1_j received from P_j in Round 1.
3. For each $k \in \mathbb{A}$ and $l \in [z+1, \mathsf{L}]$: Compute and broadcast $\overline{s}^{(k)}_{i,l} \leftarrow \mathtt{vote}(crs, \mathtt{sk}^{(k,l)}_i, \mathtt{pk}^{(k,l)}_{1\to i}, \ldots, \mathtt{pk}^{(k,l)}_{n\to i}, \nu_{i,l})$.
4. For $l \in [z+1, \mathsf{L}]$
   (a) If $\mathsf{flag} = 0$ then let $\mathtt{C}_i$ be the circuit computing $\mathtt{snd\text{-}msg}_i$ run $(\mathtt{GC}_i, \{\tilde{K}_{i,l}\}_{l\in[\mathsf{L}]}) \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}_i, 0^L)$ and send $\mathtt{abort}_j$ to the trusted third party.
   (b) Otherwise run $(\mathtt{GC}_i, \{\tilde{K}_{i,l}\}_{l\in[\mathsf{L}]}) \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}_i, \hat{\mathsf{msg}}^2_i)$.
5. For every $l \in [z+1, \ldots, \mathsf{L}]$, broadcast $s_{i,l} \leftarrow \mathtt{share}(crs, \mathtt{pk}^{(i,l)}_{1\to i}, \ldots, \mathtt{pk}^{(i,l)}_{n\to i}, K^{(0)}_{i,l}, K^{(1)}_{i,l})$, where $K^{\nu_{i,l}}_{i,l} = \tilde{K}_{i,l}$ and $K^{1-\nu_{i,l}}_{i,l}$ is chosen at random.
6. Broadcast $(\mathtt{GC}_i, \{\tilde{K}_{i,l}\}_{l\in[z]})$.
7. Broadcast $(\mathsf{msg}^1_{1\to i}, \ldots, \mathsf{msg}^1_{n\to i})$.

Output Computation. $\mathcal{S}$ on behalf of each party P_i, where $i \in \mathbb{H}$, does the following:

If there is a party who did not generate ballots for each garbled circuit based on the first-round messages that she echoed, blame that party:

1. For $j \in [n]$: Check if $\{\mathsf{msg}^1_{k\to j}\}_{k\in[n]}$ broadcast by P_j is consistent with $\{\overline{s}^{(k)}_{j,l}\}_{k\in[n], l\in[z+1,\mathsf{L}]}$. Output $\mathtt{abort}_j$ if the check fails. Else, set $(\nu_{j,z+1}, \ldots, \nu_{j,\mathsf{L}})$ as the bits comprising $(\mathsf{msg}^1_{1\to j}, \ldots, \mathsf{msg}^1_{n\to j})$.
2. For $j \in [n]$: If there does not exist a set $|S_j| \geq n-t$ such that, for $k \in S_j$, $\mathsf{msg}^1_{j\to k} = \mathsf{msg}^1_j$ holds; send $\mathtt{abort}_j$ to the trusted third party.
3. For $k \in [n]$ (whose garbled circuit we will now consider):
   (a) For each $l \in [z+1, \ldots, \mathsf{L}]$: Compute $K_{k,l} \leftarrow \mathtt{reconstruct}(crs, s_{k,l}, (\mathtt{pk}^{(k,l)}_1, v_{1,l}, \overline{s}^{(k)}_{1,l}), \ldots, (\mathtt{pk}^{(k,l)}_n, v_{n,l}, \overline{s}^{(k,l)}_{n,l})$.
      i. If $\mathtt{reconstruct}$ returns $(\perp, \mathtt{id})$, output $\mathtt{abort}_{\mathtt{id}}$.
   (b) Evaluate $\mathsf{msg}^2_k \leftarrow \mathtt{eval}(\mathtt{GC}_k, (K_{k,1}, \ldots, K_{k,\mathsf{L}}))$. If evaluation fails send $\mathtt{abort}_k$ to the trusted third party.
4. If $\mathcal{S}$ did not abort sends continue to the trusted third party.
5. $\mathcal{S}$ outputs the output of $\mathcal{A}$ and terminates.

We now define a series of hybrid experiments in order 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. $\mathtt{Expt}^0_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$: In this experiment, the simulator $\mathcal{S}_0$ has access to the internal state of the trusted party computing $\mathcal{F}$, therefore $\mathcal{S}_0$ can see the input values of honest parties and chooses the output values of the honest parties.

In the execution of $\Pi_{\mathsf{p2pbc}}^{\mathsf{id\text{-}abort}}$ 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 $\Pi_{\mathsf{p2pbc}}^{\mathsf{id\text{-}abort}}$ explained above. It follows trivially that the output of $\mathtt{Expt}^0_{\mathbb{A},\mathcal{A},\Pi_{\mathsf{p2pbc}}^{\mathsf{id\text{-}abort}}}$ and the output of the real world experiment are identically distributed.

2. $\mathtt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi_{\mathsf{p2pbc}}^{\mathsf{id\text{-}abort}}}$: In this experiment $\mathtt{Expt}^0_{\mathbb{A},\mathcal{A},\Pi_{\mathsf{p2pbc}}^{\mathsf{id\text{-}abort}}}$ 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 $i \in \mathbb{A}$. $\mathcal{S}_1$ simulates the interaction between $\mathcal{S}_{\mathtt{RS}}$ and the ideal functionality relying on the trusted third party. 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 $\mathcal{S}_{\mathtt{RS}}$.
   Let $\{\mathsf{msg}^1_{j\to i}\}_{j\in\mathbb{A}}$ be the set of messages received from $\mathcal{A}$ in Round 1, from all $i \in \mathbb{H}$. For all $j \in \mathbb{A}$, $\mathcal{S}_1$ scans the messages $\{\mathsf{msg}^1_{j\to i}\}_{i\in\mathbb{H}}$ bit by bit and $\forall\beta \in \ell$ checks what is the β-th bits b_j^β that is consistent among $(\frac{1}{3}n+1)$ of the messages $\{\mathsf{msg}^1_{j\to i}\}_{i\in\mathbb{H}}$. If it does not exist such a bit (which implies that at least $(\frac{1}{3}n+1)$ of these messages are not consistent, i.e. the adversary did not send enough first rounds messages to the honest parties) $\mathcal{S}_1$ sends $\mathtt{abort}_j$ to the trusted third party after that the second round is played (as an honest player and $\mathcal{S}$ would do).
   $\mathcal{S}_1$ executes also the same checks that the ideal world simulator $\mathcal{S}$ (described above) in steps 1, 2, 3(a)i and 3b does. If one of the checks fail $\mathcal{S}_1$ aborts identifying the cheater according to the strategy of $\mathcal{S}$ in the corresponding steps.

Claim. $\mathtt{Expt}^0_{\mathbb{A},\mathcal{A},\Pi_{\mathsf{p2pbc}}^{\mathsf{id\text{-}abort}}}$ and $\mathtt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi_{\mathsf{p2pbc}}^{\mathsf{id\text{-}abort}}}$ are computationally indistinguishable.

Proof (Sketch). In $\mathtt{Expt}^0_{\mathbb{A},\mathcal{A},\Pi_{\mathsf{p2pbc}}^{\mathsf{id\text{-}abort}}}$ the honest parties output $\mathtt{abort}_j$ if $\mathcal{A}$, on behalf of some malicious party P_j does not send consistent first-round messages of $\Pi_{\mathtt{bc}}$ to at least $(\frac{1}{3}n+1)$ honest parties. Note that if P_j does not send consistent first-round messages of $\Pi_{\mathtt{bc}}$ to at least $(\frac{1}{3}n+1)$ honest parties, then even in $\mathtt{Expt}^0_{\mathbb{A},\mathcal{A},\Pi_{\mathsf{p2pbc}}^{\mathsf{id\text{-}abort}}}$, honest parties abort identifying P_j (as in such a case, P_j's first-round message will be echoed by fewer than $(n-t)$ parties).
If the check described above did not fail, then $\mathcal{A}$ recovers the garbled 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 garbled circuits on behalf of dishonest parties. If the garbled circuit evaluation fails corresponding to garbler P_j or the ballots for the one-or-nothing secret sharing with intermediaries are generated inconsistently w.r.t. the bits of the first-round messages, honest parties abort identifying the cheater j or k respectively (as described for the honest parties and $\mathcal{S}$) in $\mathtt{Expt}^0_{\mathbb{A},\mathcal{A},\Pi_{\mathsf{p2pbc}}^{\mathsf{id\text{-}abort}}}$. Finally, we note that if the reconstruction fails (check 3(a)i of $\mathcal{S}$) both $\mathcal{S}_0$ that $\mathcal{S}_1$ can identify the cheater due

to the identifiability property of the one-or-nothing secret sharing with intermediaries. In this case in $\texttt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ one of the checks in steps 1, 2, 3(a)i and 3b of $\mathcal{S}$ fail and therefore $\mathcal{S}_1$ in $\texttt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ sends $\texttt{abort}_j$, or $\texttt{abort}_{\texttt{id}}$ or $\texttt{abort}_k$ to $\mathcal{F}$ accordingly. We conclude that honest parties aborts (identifying the correct cheater) in $\texttt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ only when the honest parties are aborting in $\texttt{Expt}^0_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$. If all the checks above did not fail then it is possible to claim that $\texttt{Expt}^0_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ and $\texttt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ are computationally indistinguishable relying on the security of $\Pi_{\texttt{bc}}$.

3. $\texttt{Expt}^2_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$: This experiment proceeds as the experiment $\texttt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ except that for the honest parties, $\mathcal{S}_2$ executes $\texttt{share}$ following the steps 5 described for $\mathcal{S}$.

   Claim. $\texttt{Expt}^2_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ and $\texttt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ are computationally indistinguishable.

   The proof proceeds via $|\mathbb{H}| + 1$ hybrids arguments: in the i-th hybrid experiment, honest party P_h with $h \leq i$ executes $\texttt{share}$ as in $\texttt{Expt}^2_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ and for $h > i$ executes $\texttt{share}$ as in $\texttt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$. It follows from the security properties of one-or-nothing secret sharing with intermediaries that two consecutive hybrids are computationally indistinguishable. In more detail, let us consider the labels $K^0_{h,v}, K^1_{h,v}$ for the garbled circuit of the honest party P_h and specifically for a wire v, where v corresponds to the input of some malicious party P_j (i.e. to the messages $\textsf{msg}^1_{j\to h}$). We analyze the following cases:
   (a) In the first round $\mathcal{A}$ (on behalf of P_j) sends first-round messages for $\Pi_{\texttt{bc}}$ s.t. more than $\frac{1}{3}n$ of those messages were consistent for the v-th bit b_v. Therefore more than half of the honest parties will run $\texttt{vote}$ w.r.t. the same bits b_v. In this case from the privacy of the one-or-nothing secret sharing with intermediaries we are guaranteed that no-information will be revealed about $K^{1-b_v}_{h,v}$ (which in $\texttt{Expt}^4_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ will correspond to the label that we are not giving as input to the simulator of the garbled circuit).
   (b) If conditions 3a does not verify i.e. less than $\frac{1}{3}n$ of the parties voted for the same bit, the privacy of one-or-nothing secret sharing with intermediaries guarantees that the adversary learns none of the labels.

   The same analysis can be conducted for the wire of the garbled circuits of party P_h.
   The proof conclude observing that the 0-th hybrid corresponds to $\texttt{Expt}^1_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ and the $|\mathbb{H}|$-th corresponds to $\texttt{Expt}^2_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$.
4. $\texttt{Expt}^3_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$: This experiment proceeds as the experiment $\texttt{Expt}^2_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ except that the garbled circuits corresponding to the honest parties are computed using the simulated procedure $\texttt{simGC}$. In more detail, $\mathcal{S}_4$ executes, for

all $h \in \mathbb{H}$, $(\mathtt{GC}_h, \{\tilde{K}_{h,l}\}_{l\in[\mathsf{L}]}) \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}_h, \mathsf{msg}^2_h)$, where msg^2_h is the message computed by P_h in the execution of $\Pi_{\mathtt{bc}}$ and $\mathtt{C}_h$ is the circuit $\mathtt{snd\text{-}msg}_h$.

Claim. $\mathtt{Expt}^2_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ and $\mathtt{Expt}^3_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ are computationally indistinguishable.

Proof (Sketch). The proof proceeds via $|\mathbb{H}|+1$ hybrids arguments: in the i-th hybrid experiment the garbled circuit of honest party P_h with $h \leq i$ are simulated as in $\mathtt{Expt}^3_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ and for $h > i$ are computed as in $\mathtt{Expt}^2_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$. 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 $\mathtt{Expt}^2_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ and the $|\mathbb{H}|$-th corresponds to $\mathtt{Expt}^3_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$.

5. $\mathtt{Expt}^4_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$: in this experiment $\mathtt{Expt}^3_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ is modified as follows. Let $\mathtt{C}_h$ be the circuit $\mathtt{snd\text{-}msg}_h$. if $\mathcal{A}$, on behalf of some malicious party j sends, to the honest parties, more than $\frac{1}{3}n$ inconsistent first round messages of $\Pi_{\mathtt{bc}}$ then $\mathcal{S}_4$ P_h (for all $h \in \mathbb{H}$) executes $(\mathtt{GC}_h, \{\tilde{K}_{h,l}\}_{l\in[\mathsf{L}]}) \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}_h, 0^L,)$ (i.e. she garbles the circuit on a dummy output); otherwise she executes $(\mathtt{GC}_h, \{\tilde{K}_{h,l}\}_{l\in[\mathsf{L}]}) \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}_h, \mathsf{msg}^2_h)$, where msg^2_h is the message computed by P_h in the execution of $\Pi_{\mathtt{bc}}$ and $\mathtt{C}_h$ is the circuit $\mathtt{snd\text{-}msg}_h$.

   Claim. $\mathtt{Expt}^3_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ and $\mathtt{Expt}^4_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ are computationally indistinguishable.

   Proof (Sketch). This proceeds via hybrid experiments similar as in the proof of Claim 4, the only extra observation is that in both in $\mathtt{Expt}^3_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ that in $\mathtt{Expt}^4_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ $\mathcal{A}$ does not learn all the labels for the garbled circuit if the $\mathcal{A}$ on behalf of some malicious party j sends more than $\frac{1}{3}n$ of inconsistent 1st round messages of $\Pi_{\mathtt{bc}}$ to the honest parties. Note that due to the partial evaluation resiliency property of the garbled circuit if $\mathcal{A}$ does not learn all the labels, $\mathcal{A}$ is not able to evaluate the garbled circuit of the honest party P_h.

6. $\mathtt{Expt}^5_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$: This experiment proceeds as the experiment $\mathtt{Expt}^4_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ except that instead of computing the messages of $\Pi_{\mathtt{bc}}$ using honest parties inputs, the simulator $\mathcal{S}_5$ uses the messages given as output by $\mathcal{S}_{\mathtt{RS}}$. In more detail: For all $h \in \mathbb{H}$ in the first round: $\mathcal{S}_5$ sends $\hat{\mathsf{msg}}^1_h$; in the second round $\mathcal{S}_5$ computes $(\mathtt{GC}_h, \{\tilde{K}_{h,l}\}_{l\in[\mathsf{L}]}) \leftarrow \mathtt{simGC}(1^\lambda, \mathtt{C}_h, \hat{\mathsf{msg}}^2_h)$ and executes $\mathtt{vote}$ w.r.t. the messages $\{\hat{\mathsf{msg}}^1_h\}_{h\in\mathbb{H}}$.

   Claim. $\mathtt{Expt}^5_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ and $\mathtt{Expt}^4_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ are computationally indistinguishable.

Proof (Sketch). In $\texttt{Expt}^5_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ the adversary learns the second messages of $\Pi_{\texttt{bc}}$ w.r.t. the honest parties only when for all $j \in \mathbb{A}$ $\mathcal{A}$ sends 1st round messages $\{\mathsf{msg}^1_{j \to i}\}_{i \in \mathbb{H}}$ of $\Pi_{\texttt{bc}}$ s.t. for each $\beta = 1, \ldots, \ell$ it is possible to identify bit b^β_j that is consistent among $\frac{1}{3}n + 1$ of messages $\{\mathsf{msg}^1_{j \to i}\}_{i \in \mathbb{H}}$. Intuitively, the first round message sent in $\Pi_{\texttt{bc}}$ by P_j, for all $j \in \mathbb{A}$, corresponds to the string $b^1_j, \ldots, b^n_j$. Therefore, the indistinguishability between $\texttt{Expt}^4_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ and $\texttt{Expt}^5_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ follows from the security of $\Pi_{\texttt{bc}}$. In more detail, any distinguisher between $\texttt{Expt}^5_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ and $\texttt{Expt}^4_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ can be used to distinguish between $\mathcal{S}_{\texttt{RS}}$ and a real execution with the receiver-specific adversary. As observed in [CGZ20], the proof crucially rely on the ability of the simulator $\mathcal{S}_{\texttt{RS}}$ to extract the adversary's input from her first round of $\Pi_{\texttt{bc}}$.

The proof ends observing that in $\texttt{Expt}^5_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ $\mathcal{S}_5$ does not need anymore to have access to the internal state of the trusted third party that computes $\mathcal{F}$ and therefore $\texttt{Expt}^5_{\mathbb{A},\mathcal{A},\Pi^{\text{id-abort}}_{\text{p2pbc}}}$ and the ideal world experiment are identically distributed.

E Proof of Theorem 3

Consider protocol Π which is the same as $\Pi_{\texttt{bc}}$, except that the second round is sent over peer-to-peer channels. Intuitively, the only advantage that the adversary has in Π is to send inconsistent last-round messages. However, we argue that this cannot lead to a pair of honest parties obtaining two different non-$\bot$ outputs. This is because of our assumption that the simulator of $\Pi_{\texttt{bc}}$ extracts input from the first round messages (and say receives the output y from the ideal functionality).

A bit more formally, let us describe how the simulator $\mathcal{S}$ of Π works. Let us denote with $\mathcal{S}_{\texttt{bc}}$ the simulator for $\Pi_{\texttt{bc}}$, and with $\mathbb{A}$ the set of dishonest parties. $\mathcal{S}$ runs internally $\mathcal{S}_{\texttt{bc}}$ emulating for her an adversary $\mathcal{A}_{\texttt{bc}}$ of $\Pi_{\texttt{bc}}$ (and the ideal functionality) while interacting as the honest parties w.r.t. the adversary $\mathcal{A}$ in the execution of Π; details follow. $\mathcal{S}$ starts $\mathcal{S}_{\texttt{bc}}$ which gives the first round messages of $\Pi_{\texttt{bc}}$ for the honest parties and $\mathcal{S}$ forwards them to $\mathcal{A}$ in the execution of Π. Upon receiving the first rounds of Π from $\mathcal{A}$, $\mathcal{S}$ forwards them to $\mathcal{S}_{\texttt{bc}}$. At this point, by theorem's assumption, the simulator $\mathcal{S}_{\texttt{bc}}$ extracts the inputs of the corrupted parties (or sends an abort identifying some corrupted parties) and queries the ideal functionality which is emulated by $\mathcal{S}$. $\mathcal{S}$ queries the (real) ideal functionality forwarding the message of $\mathcal{S}_{\texttt{bc}}$, obtaining the output y in case an abort did not occur. If an abort did not occur the simulation continues as follows. $\mathcal{S}$ sends y to $\mathcal{S}_{\texttt{bc}}$ obtaining the second round for the honest parties which is played in the execution of Π. At this point, in the execution of Π, that is over a peer-to-peer channel, the adversary on behalf of the corrupted party P_j, for all $j \in \mathbb{A}$, can have two different behavior w.r.t. different honest parties:

1) P_j sends a malformed second round (or no second round at all) to the honest party P_h, in this case, $\mathcal{S}$ (on behalf of the honest party) can efficiently detect the invalidity of the second round and query the ideal functionality with $\texttt{abort}_j$ for party P_h.

Observe that $\mathcal{S}$ has to check the validity of the second round of P_j only when P_h receives second round messages from all the other parties, indeed if P_h did not receive second round messages from some of the parties, she aborts blaming the parties that did not send her the message.

2) P_j sends a well-formed second round to the honest party $P_{h'}$, in this case, $\mathcal{S}$ instructs the ideal functionality to let $P_{h'}$ recover the output. $\mathcal{S}$ terminates giving as output the view of $\mathcal{A}$.

The indistinguishability between the real game and the simulated game follows from the security of Π_{bc}.

It remains to show a BC-BC protocol (respectively a P2P-BC) with identifiable abort security that satisfies the properties of Theorem 3. It was already proved in [CGZ20] that the 2-round BC-BC protocol, in the CRS model, described in [BL18] satisfies the identifiable abort property. The protocol described in [BL18] uses non-interactive zero-knowledge (NIZK) proof of knowledge to compile a semi-malicious protocol into a fully malicious one in each round one. Therefore, it is possible to extract the adversarial inputs in the first rounds of the protocol, and the NIZK that accompanies the second round can be used to check the validity of the second round.

Finally, we observe that it is possible to compile the P2P-BC protocol $\Pi^{\mathsf{id\text{-}abort}}_{\mathsf{p2pbc}}$ described in Figure 3.2 using the BC-BC protocol described in [BL18]. In this way the protocol $\Pi^{\mathsf{id\text{-}abort}}_{\mathsf{p2pbc}}$ (Theorem 2), will satisfy the first property stated in Theorem 3. The second property, instead, follow from the possibility of efficiently checking the second round message of the underlying BC-BC protocol and from the fact that the steps 1,2,3 in Figure 3.2 also are efficiently checkable given the second round messages of all the other parties. Similarly, it can be argued that the P2P-BC, IA, PKI, $t < n/2$ construction of [DMR$^+$21] also satisfies both required properties.

F Impossibility of P2P-BC, SIA with PKI when $t < n$

Observation 1 (P2P-BC, SIA, PKI, $n > t$) *There exist functions f such that no n-party two-round protocol can compute f with selective identifiable abort against $t < n$ corruptions while making use of broadcast only in the second round (i.e. where the first round is over peer-to-peer channels and the second round uses both broadcast and peer to-peer channels).*

We observe that the impossibility result of P2P-BC, IA, PKI $n > t$ protocol in [CGZ20] extends to SIA as well. For the sake of completeness, we give a high-level overview of their argument below and refer to [CGZ20] for the formal details.

Towards a contradiction, assume a 3-party P2P-BC, IA, PKI protocol Π that is secure against at most $t \leq 2$ corruptions. The proof in [CGZ20] considers the following three scenarios – **Scenario 1** involves a corrupt P_3 who drops her first round peer-to-peer message towards P_2 and pretends in the second-round over broadcast that she never received any first-round peer-to-peer message from P_2. **Scenario 2** is a dual version of Scenario 1 i.e. it involves a corrupt P_2 who drops her first round peer-to-peer message towards P_3 and pretends in the second-round over broadcast that she never received any first-round peer-to-peer message from P_3. **Scenario 3** involves an adversary who corrupts both P_1 and P_2 and follows the same strategy as Scenario 2 while behaving honestly on behalf of P_1.

The proof in [CGZ20] is a three-step argument – **(1)** First, they show that the protocol cannot abort in the first two scenarios, because the output distributions in the two scenarios are identically distributed (since the views of parties are identically distributed in all the scenarios) but the identity of the cheaters is different. In some more detail, if Scenario 1 outputs P_3 as cheater with noticeable probability, then P_3 would be output as cheater in Scenario 2 as well with noticeable probability (which is unacceptable as P_3 is honest in Scenario 2). Therefore, it must be the case that the parties do not abort in Scenarios 1 and 2. **(2)** The next observation is that in Scenario 3, despite P_1 being corrupt, since she behaved honestly with respect to some input, say x_1, the output obtained by honest parties must be on the input x_1. **(3)** The last part of the argument shows that if the adversary in Scenario 3 is rushing, she could obtain P_3's second round message before sending any message on behalf of P_2. Since P_3's second round message is independent of P_2's input, the adversary can emulate in her head two executions of the protocol using the same messages for P_3 and same first-round message for P_1 but different inputs for P_2. The proof shows that this attack violates input privacy of honest P_3 as the adversary learns more information about P_3's input than allowed by the ideal evaluation with respect to a carefully designed function f (we refer to [CGZ20] for further details). This contradicts the security of Π.

We note that the above argument extends to SIA – Similar to **(1)**, we can argue that since the view of honest party P_1 is identically distributed in Scenarios 1 and 2; her view must not lead her to identify either P_2 or P_3 as cheater with noticeable probability (as she would be wrong in one of the two cases). Therefore, to satisfy SIA guarantee with respect to P_1, we can conclude that both Scenarios 1 and 2 lead to P_1 obtaining the output with overwhelming probability. Next, the same observation **(2)** would show that the output obtained by P_1 (who behaved honestly) in Scenario 3 must be computed on x_1. Lastly, the same attack as outlined in **(3)** can be carried out, allowing the adversary to emulate multiple executions in her head with respect to different choices of input on behalf of P_2 and obtaining multiple evaluations of the function (via the output of P_1), which contradicts security.