

# On the Round Complexity of Fully Secure Solitary MPC with Honest Majority

Saikrishna Badrinarayanan<sup>1</sup>, Peihan Miao<sup>2</sup>, Pratyay Mukherjee<sup>1</sup>, and Divya Ravi<sup>3</sup>

<sup>1</sup> Visa Research, {sabadrin, pratmukh}@visa.com

<sup>2</sup> University of Illinois at Chicago, peihan@uic.edu

<sup>3</sup> Aarhus University, divya@cs.au.dk

## Abstract

We study the problem of secure multiparty computation for functionalities where only *one* party receives the output, to which we refer as *solitary MPC*. Recently, Halevi et al. (TCC 2019) studied fully secure (i.e., with guaranteed output delivery) solitary MPC and showed impossibility of such protocols for certain functionalities when there is no honest majority among the parties.

In this work, we study fully secure solitary MPC in the honest majority setting and focus on its round complexity. We note that a broadcast channel or public key infrastructure (PKI) setup is necessary for an  $n$ -party protocol against malicious adversaries corrupting up to  $t$  parties where  $n/3 \leq t < n/2$ . Therefore, we study the following settings and ask the question: Can fully secure solitary MPC be achieved in fewer rounds than fully secure standard MPC in which all parties receive the output?

- When there is a broadcast channel and no PKI:
  - We start with a negative answer to the above question. In particular, we show that the exact round complexity of fully secure solitary MPC is 3, which is the same as fully secure standard MPC.
  - We then study the minimal number of broadcast rounds needed in the design of round-optimal fully secure solitary MPC. We show that both the first and second rounds of broadcast are necessary when  $2\lceil n/5 \rceil \leq t < n/2$ , whereas pairwise-private channels suffice in the last round. Notably, this result also applies to fully secure standard MPC in which all parties receive the output.
- When there is a PKI and no broadcast channel, nevertheless, we show more positive results:
  - We show an upper bound of 5 rounds for any honest majority. This is superior to the super-constant lower bound for fully secure standard MPC in the exact same setting.
  - We complement this by showing a lower bound of 4 rounds when  $3\lceil n/7 \rceil \leq t < n/2$ .
  - For the special case of  $t = 1, n = 3$ , when the output receiving party does not have an input to the function, we show an upper bound of 2 rounds, which is optimal. When the output receiving party has an input to the function, we show a lower bound of 3, which matches an upper bound from prior work.
  - For the special case of  $t = 2, n = 5$ , we show a lower bound of 3 rounds (an upper bound of 4 follows from prior work).

All our results also assume the existence of a common reference string (CRS) and pairwise-private channels. Our upper bounds use a decentralized threshold fully homomorphic encryption (dTFHE) scheme (which can be built from the learning with errors (LWE) assumption) as the main building block.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Our Results . . . . .	5
1.2	Roadmap . . . . .	7
<b>2</b>	<b>Technical Overview</b>	<b>8</b>
2.1	Overview of Upper Bounds . . . . .	8
2.2	Overview of Lower Bounds . . . . .	10
<b>3</b>	<b>Preliminaries</b>	<b>13</b>
3.1	Notation and Setting . . . . .	13
3.2	Security Model . . . . .	13
3.3	Cryptographic Primitives . . . . .	14
<b>4</b>	<b>With Broadcast and No PKI</b>	<b>18</b>
4.1	Necessity of Three Rounds . . . . .	18
4.2	Necessity of Broadcast in Round 1 . . . . .	22
4.3	Necessity of Broadcast in Round 2 . . . . .	25
4.4	Upper Bounds . . . . .	30
<b>5</b>	<b>With PKI and No Broadcast</b>	<b>32</b>
5.1	Necessity of Four Rounds . . . . .	32
5.2	General Five-Round Protocol . . . . .	38
5.3	$(t + 2)$ Round Protocol . . . . .	41
<b>6</b>	<b>Special Case: <math>t = 1</math> with PKI and no Broadcast</b>	<b>41</b>
6.1	Necessity of Three Rounds When $Q$ Has Input . . . . .	41
6.2	Protocols . . . . .	43
<b>7</b>	<b>Special Case: <math>t = 2</math> with PKI and no Broadcast</b>	<b>45</b>
7.1	Necessity of Three Rounds . . . . .	45
7.2	Four-Round Protocol . . . . .	48
<b>A</b>	<b>Literature Survey</b>	<b>54</b>
<b>B</b>	<b>Necessity of Broadcast/PKI</b>	<b>55</b>
<b>C</b>	<b>Security Proof For Five-Round Protocol</b>	<b>56</b>
<b>D</b>	<b>Security Proof for <math>t = 1</math> When <math>Q</math> Has No Input</b>	<b>63</b>

# 1 Introduction

Secure multiparty computation (MPC) [Yao86,GMW87] allows a set of mutually distrusting parties to jointly compute any function on their private data in a way that the participants do not learn anything about the inputs except the output of the function. The strongest possible security notion for MPC is *guaranteed output delivery* (**god** for short), which states that all honest parties are guaranteed to receive their outputs no matter how the corrupt parties behave. An MPC protocol achieving **god** is often called a *fully secure* protocol. A seminal work of Cleve [Cle86] showed that there exist functionalities for which it is impossible to construct an MPC protocol with **god** unless a majority of the parties are honest.

**Solitary MPC.** Recently, Halevi et al. [HIK<sup>+</sup>19] initiated the study of MPC protocols with **god** for a special class of functionalities, called *solitary* functionalities, which deliver the output to *exactly one party*. Such functionalities capture many real world applications of MPC in which parties play different roles and only one specific party wishes to learn the output. For example, consider a privacy-preserving machine learning task [MZ17] where several entities provide training data while only one entity wishes to learn a model based on this private aggregated data. As another example, a service provider may want to learn aggregated information about its users while keeping the users' data private [BIK<sup>+</sup>17,BBG<sup>+</sup>20]. In the rest of the paper we refer to such MPC protocols as *solitary MPC*. For clarity of exposition, we refer to protocols where all parties obtain output as *standard MPC*. While the argument of Cleve [Cle86] does not rule out solitary MPC with **god** in the presence of a dishonest majority,<sup>1</sup> Halevi et al. [HIK<sup>+</sup>19] showed that there exist functionalities for which solitary MPC with **god** is also impossible with dishonest majority. Hence, the results of [Cle86] and [HIK<sup>+</sup>19] rule out the existence of a generic MPC protocol that can compute *any* standard and solitary functionality respectively with **god** in dishonest majority (protocols can exist for specific classes of functionalities as shown in [GHKL11,ABMO15,HIK<sup>+</sup>19]). Both impossibility results hold even when parties have access to a common reference string (CRS). In this paper, we focus on *solitary MPC with god* in the *honest majority* setting.

**Round Complexity.** An important efficiency metric of an MPC protocol is its *round complexity*, which quantifies the number of communication rounds required to perform the protocol. The round complexity of standard MPC has been extensively studied over the last four decades (see Appendix A for a detailed literature survey). In the honest majority setting, *three* rounds are known to be necessary [GIKR02,PR18,GLS15] for *standard MPC with god*, even in the presence of a common reference string (CRS) *and* a broadcast channel (without a PKI setup). Matching upper bounds appear in [GLS15,ACGJ18,BJMS20]. The protocol of Gordon et al. [GLS15] requires a CRS, while the other two [ACGJ18,BJMS20] are in the plain model. In this work we focus on the round complexity aspects of solitary MPC protocols.

**Necessity of Broadcast or PKI.** A closer look at the above protocols reveals that all of them assume the existence of a broadcast channel. For solitary MPC with **god**, the works of

---

<sup>1</sup>Cleve's argument shows that with dishonest majority, it is impossible for an MPC protocol to achieve *fairness*, which guarantees that malicious parties cannot learn the output while preventing honest parties from learning the output. Since **god** implies fairness, this impossibility also holds for standard MPC with **god**. However, it doesn't hold for solitary MPC as fairness is clearly not an issue in the solitary MPC setting.

[FGMO01, ACOS20] show that either a broadcast channel or a public key infrastructure (PKI) setup is indeed necessary assuming an honest majority (in particular, when  $n/3 \leq t < n/2$  for an  $n$ -party protocol against adversaries corrupting up to  $t$  parties) even with a CRS.<sup>2</sup> Note that although PKI setup and broadcast channels are equivalent according to [DS83] from a feasibility perspective, realizing broadcast under PKI setup with *guaranteed termination* requires super-constant rounds, which we will discuss shortly. In light of this, we study the round complexity of solitary MPC with `god` when  $n/3 \leq t < n/2$  in two settings: (a) there is a broadcast channel and no PKI setup; (b) there is PKI setup and no broadcast channel. When both broadcast channels and PKI are available, we know from prior works [HLP11, GLS15] that the exact round complexity is two.

**With Broadcast, No PKI.** In this setting we investigate whether we can do better for solitary MPC than standard MPC in terms of round complexity even in the presence of CRS. In particular,

*Assuming a broadcast channel and CRS, can we build a solitary MPC protocol with `god` in fewer than three rounds?*

Unfortunately, the answer is no! We show that in the presence of a broadcast channel and CRS, the exact round complexity for solitary MPC with `god` is also *three*, same as standard MPC.

However, broadcast channels are *expensive* to realize in practice – the seminal works of Dolev and Strong [DS83] and Fischer and Lynch [FL82] showed that realizing a single round of *deterministic* broadcast requires at least  $t + 1$  rounds of communication over pairwise-private channels, where  $t$  is the number of corrupt parties, even with a public key infrastructure (PKI) setup.<sup>3</sup> This can be overcome by considering *randomized* broadcast protocols in the honest majority setting [FM89, FG03, KK09, ADD<sup>+</sup>19] requiring expected constant rounds. In particular, the most round-efficient protocol to our knowledge is proposed by Abraham et al. [ADD<sup>+</sup>19], which solves Byzantine agreement for  $t < n/2$  in expected 10 rounds. Nevertheless, these protocols do not *guarantee termination* in constant rounds, which is the setting we are interested in.<sup>4</sup> In fact, it is shown that termination cannot be guaranteed in constant rounds [CMS89, KY86].

Recent works [GGJ19, CGZ20, DMSY20] try to minimize the usage of expensive broadcast channels in the context of round-optimal standard MPC. In particular, they study whether each round of a round-optimal MPC protocol necessarily requires a broadcast channel or pairwise-private channels suffice in some of them. In the context of round-optimal solitary MPC with `god`, we ask an analogous question:

*Is a broadcast channel necessary in every round of a three-round solitary MPC protocol with `god`?*

We show that a broadcast channel is *necessary* in both the *first* and *second* rounds in a three-round solitary MPC protocol with `god` while pairwise-private channels suffice in the third round.

<sup>2</sup>Fitzi et al. [FGMO01] show that converge-cast cannot be achieved when  $n/3 \leq t < n/2$  in the *information theoretic* setting. Alon et al. [ACOS20] show a specific solitary functionality that cannot be computed by a 3-party MPC protocol with a single corruption with `god` in the *plain model* (with no broadcast channel and no PKI), which also extends to  $n/3 \leq t < n/2$ . Both arguments also work even in the presence of a CRS. We present the proof in [Appendix B](#) for completeness.

<sup>3</sup>Note that PKI setup is in fact necessary for realizing a broadcast channel when  $t \geq n/3$  (where  $n$  is the total number of parties) [PSL80, LSP82].

<sup>4</sup>In these randomized broadcast protocols, the number of rounds depends on the randomness involved in the protocol. For example, the protocol by Abraham et al. [ADD<sup>+</sup>19] terminates in constant rounds except with constant probability and requires at least super-polylogarithmic rounds (in the security parameter) to terminate with all but negligible probability.

**With PKI, No Broadcast.** In this setting a natural question arises: in the absence of a broadcast channel, if we assume a PKI setup, what is the optimal round complexity for solitary MPC with `god`? In standard MPC, note that since standard MPC with `god` implies broadcast with guaranteed termination, any protocol without a broadcast channel (only using pairwise-private channels with PKI setup) should *necessarily* require super-constant rounds. In contrast, observe that solitary MPC with `god` does not imply broadcast with guaranteed termination, so the same lower bound does not hold. This motivates us to ask the following question:

*With a PKI setup and no broadcast channel, can we overcome the above standard MPC lower bound? Specifically, can we build a constant-round solitary MPC protocol with `god` in the honest majority setting?*

## 1.1 Our Results

### 1.1.1 With Broadcast, No PKI

When there is a broadcast channel but no PKI setup, we show a lower bound of *three rounds* for achieving solitary MPC with `god` in the honest majority setting, which is the same as the lower bound for standard MPC.

**Informal Theorem 1.** *Assume parties have access to CRS, pairwise-private channels and a broadcast channel. Then, there exists a solitary functionality  $f$  such that no two-round MPC protocol can compute  $f$  with `god` in the honest majority setting (in particular, when  $n/3 \leq t < n/2$ ) even against a non-rushing adversary.*

This lower bound is tight because we know from prior works [GLS15, ACGJ18, BJMS20] that there are three-round solitary MPC protocols with `god` in the honest majority setting.

We then study the minimal number of broadcast rounds needed in a round-optimal (three-round) solitary MPC protocol with `god`. We show that a broadcast channel is necessary in both the first and second rounds.

**Informal Theorem 2.** *Assume parties have access to CRS and pairwise-private channels. No three-round solitary MPC protocol can compute any solitary functionality  $f$  with `god` in the honest majority setting (in particular, when  $2 \lceil n/5 \rceil \leq t < n/2$ ) even against a non-rushing adversary, unless there are broadcast channels in both Rounds 1 and 2.*

We note that the necessity of a broadcast channel in Round 1 holds for any  $n/3 \leq t < n/2$  while the necessity of a broadcast channel in Round 2 only holds for  $2 \lceil n/5 \rceil \leq t < n/2$  requiring *at least two parties be corrupted*. In other words, for  $t = 1$  and  $n = 3$  only the first round broadcast is necessary. This is consistent with and proven tight by the upper bound in the work of Patra and Ravi [PR18], which constructed a three-round three-party protocol with `god` tolerating a single corruption, using broadcast only in Round 1.

For the general case when  $t \geq 2$ , we observe that in the three-round protocols from prior work [GLS15, ACGJ18, BJMS20], only the first two rounds require a broadcast channel while the third-round messages can be sent over pairwise-private channels to the output-receiving party. Thus, our lower bounds are also tight in the general case.

**Implications for Standard MPC.** The work of Cohen et al. [CGZ20] identifies which rounds of broadcast are necessary for achieving round-optimal (two-round) standard MPC with *dishonest majority*. The recent work of [DMSY20] studies this question for two-round standard MPC in the honest majority setting, assuming the presence of a correlated randomness setup (or PKI). However, the same question for round-optimal (*three-round*) standard MPC with *god* in *honest majority* setting and without correlated randomness (or PKI) is not known; which we address in this work. Since standard MPC with *god* implies solitary MPC with *god* (via a generic transformation), our negative results for solitary MPC also apply to standard MPC, namely both the first and second rounds of broadcast are necessary for a three-round standard MPC with *god*. On the other hand, we observe that the existing three-round protocols [GLS15, BJMS20] still work if the third-round messages are sent over pairwise-private channels (see the discussion in Section 4.4), thus we fully resolve this problem for standard MPC with *god* in honest majority setting and without correlated randomness setup (i.e., in the plain and CRS models).

### 1.1.2 With PKI, No Broadcast

When there is a PKI setup and no broadcast channel, we show that the super-constant lower bound for standard MPC does *not* hold for solitary MPC any more. In particular, we construct a *five-round* protocol that works for any number of parties and achieves *god* in the honest majority setting. Our protocol builds on the standard MPC protocol with *god* of Gordon et al. [GLS15] and uses a decentralized threshold fully homomorphic encryption (dTFHE) scheme (defined in [BGG<sup>+</sup>18]) as the main building block, which can be based on the learning with errors (LWE) assumption. Our PKI setup includes a setup for digital signatures as well as one for dTFHE (similarly as in [GLS15]).

**Informal Theorem 3.** *Assuming LWE, there exists a five-round solitary MPC protocol with *god* in the presence of PKI and pairwise-private channels. The protocol works for any number of parties  $n$ , any solitary functionality and is secure against a malicious rushing adversary that can corrupt any  $t < n/2$  parties.*

We complement this upper bound by providing a lower bound of *four rounds* in the same setting even in the presence of a non-rushing adversary.

**Informal Theorem 4.** *Assume a PKI setup and pairwise-private channels. There exists a solitary functionality  $f$  such that no three-round MPC can compute  $f$  with *god* in the honest majority (in particular, when  $3 \lceil n/7 \rceil \leq t < n/2$ ) even against a non-rushing adversary.*

The above lower bound requires  $t \geq 3$ , namely at least 3 parties are corrupted. Separately we also study the round complexity for scenarios when  $t < 3$ .

**Special case:  $t = 1$ .** When there is only 1 corrupted party, the only relevant setting is when  $n = 3$ . We consider two cases: (a) when the function  $f$  involves an input from the output-receiving party  $Q$ , and (b) when  $f$  does not involve an input from  $Q$ . In the first case, we show a lower bound of *three rounds* for achieving solitary MPC with *god*. That is, there exists a solitary functionality  $f$  (involving an input from  $Q$ ) such that a minimum of three rounds are required to achieve solitary MPC with *god*. Notably, this lower bound also extends to any  $n \geq 3$  and  $n/3 \leq t < n/2$ . A three-round upper bound for  $t = 1$  can be achieved by combining [GLS15] and [DS83].

In the second case where  $f$  does not involve an input from  $Q$ , it turns out we can do better than three rounds. In particular, we show a *two-round* protocol to achieve solitary MPC with *god*.

Once again, the main technical tool is decentralized threshold FHE and the protocol can be based on LWE. This upper bound is also tight as we know from prior work [HLP11] that two rounds are necessary.

**Special case:  $t = 2$ .** When the number of corrupted parties is 2, we only consider the case of  $n = 5$  and show a lower bound of *three rounds* to compute any function  $f$  (with or without input from  $Q$ ). This lower bound also extends to any  $n \geq 5$  and  $2 \lceil n/5 \rceil \leq t < n/2$ . An upper bound of four rounds for  $t = 2$  can also be achieved by combining [GLS15] and [DS83].

We remark that all our lower bounds above hold not only for PKI, but naturally extend to arbitrary correlated randomness setup model. We summarize all our results along with the known related results for the round complexity of solitary MPC with `god` in Tables 1 and 2. Note that for certain ranges of  $(n, t)$  such as  $3 \lceil n/7 \rceil \leq t < n/2$ , it is not meaningful for every  $n$  (e.g., when  $n = 8$ , there is no appropriate  $t$  in the range). This is an artifact of the partitioning technique used in the proof. Nevertheless, the range is relevant for sufficiently large values of  $n$ . All our results also assume the existence of a common reference string (CRS) and pairwise-private channels. Our results are highlighted in red.

broadcast	PKI	$(n, t)$	$Q$ has input	lower bound	upper bound
yes	yes	$t < n/2$	—	2 [HLP11]	2 [GLS15]
yes	no	$n/3 \leq t < n/2$	—	3 (Theorem 4.1)	3 [GLS15, ACGJ18, BJMS20]
no	yes	$n = 3, t = 1$	no	2 [HLP11]	2 (Theorem 6.4)
no	yes	$n = 3, t = 1$	yes	3 (Theorem 6.1)	3 [GLS15] + [DS83]
no	yes	$n = 5, t = 2$	—	3 (Theorem 7.1)	4 [GLS15] + [DS83]
no	yes	$3 \lceil n/7 \rceil \leq t < n/2$	—	4 (Theorem 5.1)	5 (Theorem 5.7)

Table 1: Round complexity of solitary MPC with `god`. “—” means it doesn’t matter what value to take. Our results are highlighted in red.

bc in R1	bc in R2	bc in R3	$(n, t)$	Possible?
no	yes	yes	$n/3 \leq t < n/2$	No (Theorem 4.4)
yes	no	yes	$2 \lceil n/5 \rceil \leq t < n/2$	No (Theorem 4.7)
yes	yes	no	$t < n/2$	Yes [GLS15, ACGJ18, BJMS20]
yes	no	no	$n = 3, t = 1$	Yes [PR18]

Table 2: For the setting with broadcast channels and no PKI setup, we study the possibility of achieving a three-round solitary MPC with `god` with fewer broadcast rounds. “bc in R1” means the parties have access to the broadcast channel in Round 1. All parties have access to pairwise-private channels in all rounds. For all the results, it doesn’t matter whether  $Q$  has input or not. Our results are highlighted in red.

## 1.2 Roadmap

We provide an elaborated technical overview next in Section 2. We provide detailed preliminaries including the definitions of required cryptographic building blocks in Section 3. In Section 4 we present our results assuming a broadcast channel but no PKI setup. In Section 5 we provide our lower bounds for PKI without broadcast as well as our main five-round protocol as an upper bound.

In [Section 6](#) and [Section 7](#) we detail our results for  $t = 1$  and  $t = 2$  respectively. The detailed literature survey appears in [Appendix A](#).

## 2 Technical Overview

### 2.1 Overview of Upper Bounds

In this section, we give a technical overview of the upper bounds. We will mainly focus on the general five-round protocol in the setting with PKI and no broadcast, and briefly discuss other special cases at the end.

Our starting point is the two-round protocol of Gordon et al. [[GLS15](#)] which achieves guaranteed output delivery (god) in the presence of an honest majority and delivers output to all parties, assuming the existence of a broadcast channel and PKI setup. The protocol uses a  $(t + 1)$ -out-of- $n$  decentralized threshold fully homomorphic encryption (dTFHE) scheme, where an FHE public key  $\text{pk}$  is generated in the setup and the secret key is secret shared among the parties. The encryptions can be homomorphically evaluated and can only be jointly decrypted by at least  $(t + 1)$  parties. Their two-round protocol in the broadcast model roughly works as follows. First, the PKI setup generates the dTFHE public key  $\text{pk}$  and individual secret keys  $\text{sk}_i$  for each party  $P_i$ . In Round 1, each party  $P_i$  computes an encryption of its input  $x_i$  and broadcasts  $\llbracket x_i \rrbracket$ .<sup>5</sup> Then each party can homomorphically evaluate the function  $f$  on  $\llbracket x_1 \rrbracket, \dots, \llbracket x_n \rrbracket$  to obtain an encryption of the output  $\llbracket y \rrbracket$ . In Round 2, each party broadcasts a partial decryption of  $\llbracket y \rrbracket$ . At the end of this, every party can individually combine the partial decryptions to learn the output  $y$ .

One immediate observation is that since we only care about one party  $P_n (= Q)$  receiving the output, the second round also works without a broadcast channel by requiring every party to only send partial decryptions directly to  $Q$ . The main challenge now is to emulate the first round with pairwise-private channels instead of broadcast channels. A naïve approach is to employ a  $(t + 1)$ -round protocol to realize the broadcast functionality over pairwise-private channels [[DS83](#)], but this would result in a  $(t + 2)$ -round protocol.

Even worse, there seems to be a fundamental barrier in this approach to design a constant round protocol. At a high level, to achieve guaranteed output delivery, we want all the honest parties to agree on a set of ciphertexts  $\llbracket x_1 \rrbracket, \dots, \llbracket x_n \rrbracket$  so that they can homomorphically evaluate on the same set of ciphertexts and compute partial decryptions on the same  $\llbracket y \rrbracket$ . This already implies Byzantine agreement, which requires at least  $(t + 1)$  rounds [[DS83](#)].

**Circumventing the lower bound.** A crucial observation here, which also separates solitary MPC from standard MPC, is that we do not need all the honest parties to *always* agree. Instead, we need them to agree *only when  $Q$  is honest*. In other words, if the honest parties detect any dishonest behavior of  $Q$ , they can simply abort. This does not imply Byzantine agreement now. Hence there is a hope to circumvent the super-constant lower bound.

**Relying on honest  $Q$ .** First, consider a simple case where honest parties only need to agree on  $\llbracket x_n \rrbracket$  when  $Q$  is honest. This can be done in two rounds (by augmenting the two-round broadcast with abort protocol of [[GL05](#)] with digital signatures). In Round 1,  $Q$  sends  $\llbracket x_n \rrbracket$  to each party (along with its signature). To ensure  $Q$  sends the same ciphertext to everyone, in Round 2, parties

---

<sup>5</sup>We use  $\llbracket x \rrbracket$  to denote a dTFHE encryption of  $x$ .

exchange their received messages in Round 1. If there is any inconsistency, then they detect dishonest behavior of  $Q$ , so they can abort; otherwise, all the honest parties will agree on the same  $\llbracket x_n \rrbracket$  at the end of Round 2 if  $Q$  is honest. Unfortunately this simple approach does not work for parties other than  $Q$ . If honest parties want to agree on  $\llbracket x_i \rrbracket$  for  $i \neq n$ , they cannot simply abort when detecting inconsistent messages from  $P_i$  (because they are only allowed to abort when  $Q$  is dishonest).

Our next attempt is to crucially rely on  $Q$  to send out all the ciphertexts. In Round 1, each party  $P_i$  first sends an encryption  $\llbracket x_i \rrbracket$  to  $Q$ . Then in Round 2,  $Q$  sends  $\llbracket x_1 \rrbracket, \dots, \llbracket x_n \rrbracket$  to each party. In Round 3, parties exchange their messages received from  $Q$ . If the honest parties notice any inconsistency in  $Q$ 's Round-2 messages, they can simply abort. Note that every message is sent along with the sender's signature, so a malicious  $Q$  cannot forge an honest  $P_i$ 's ciphertext  $\llbracket x_i \rrbracket$ ; similarly, a malicious  $P_i$  cannot forge an honest  $Q$ 's Round-2 message. Therefore, all the honest parties will agree on the same set of ciphertexts at the end of Round 3 if  $Q$  is honest.

Nevertheless, a malicious  $Q$  has complete freedom to discard any honest party's input in Round 2 (pretending that these parties did not communicate to him in Round 1) and learn a function excluding these honest parties' inputs, which should not be permitted. The crux of the issue is the following: Even when  $Q$  is malicious, the output of  $f$  learned by  $Q$  must be either  $\perp$  or include every honest party's input. This is implied by the security guarantees of the MPC protocol. In particular, in the real/ideal paradigm, a malicious  $Q$  in the ideal world can only obtain an output from the ideal functionality that computes  $f$  involving all the honest parties' inputs. Therefore, we need a mechanism to ensure that all the honest parties' ciphertexts are picked by  $Q$ . However, the parties do not know the identities of the honest parties. How can they ensure this?

**Innocent until proven guilty.** Our solution to this problem is for every party  $P_i$  to treat other parties with more leniency. That is, unless  $P_i$  knows with absolute certainty that another party  $P_k$  is malicious,  $P_i$  would demand that the ciphertexts picked by  $Q$  must also include a ciphertext from  $P_k$ . To implement this mechanism, we add another round at the beginning, where each party  $P_i$  sends  $\llbracket x_i \rrbracket$  to every other party. Then in Round 2, each party  $P_i$ , besides sending  $\llbracket x_i \rrbracket$  to  $Q$ , also sends all the ciphertexts he has received to  $Q$ . In Round 3,  $Q$  picks a set of ciphertexts  $\llbracket x_1 \rrbracket, \dots, \llbracket x_n \rrbracket$  and sends to each party. In particular, for each party  $P_k$ , as long as  $Q$  received any valid ciphertext for  $P_k$  (either directly from  $P_k$  or from other parties),  $Q$  must include a ciphertext for  $P_k$ . Parties exchange messages in Round 4 to check  $Q$ 's consistency as before. Finally, we maintain the following invariant for every honest party  $P_i$  before sending the partial decryption in Round 5: if  $P_i$  received a ciphertext  $\llbracket x_k \rrbracket$  from party  $P_k$  in Round 1, then the ciphertexts picked by  $Q$  must also include a ciphertext from  $P_k$ . Crucially, this invariant allows  $Q$  to pick a different ciphertext  $\llbracket x'_k \rrbracket$  (with a valid signature) if e.g. that was received by  $Q$  from  $P_k$ . On the other hand, this prevents the attacks discussed earlier as a malicious  $Q$  can no longer discard an honest  $P_k$ 's ciphertext  $\llbracket x_k \rrbracket$ , although  $P_i$  is yet to identify the honest parties.

**Achieving fully malicious security.** To achieve fully malicious security, we still need to ensure that the adversary's messages are correctly generated. The approach taken by [GLS15] is to apply a generic round-preserving compiler [AJL<sup>+</sup>12] that transforms a semi-malicious protocol (where, the semi-malicious adversary needs to follow the protocol specification, but has the liberty to decide the input and random coins in each round) to a malicious protocol using non-interactive zero-knowledge (NIZK) proofs in the CRS model with broadcast channels. In particular, in each round,

the adversary must prove (in zero-knowledge) that it is following the protocol consistently with some setting of random coins. However, we cannot directly apply this round-preserving compiler since we do not have broadcast channels. This limitation introduces additional complications in our protocol design to preserve the round complexity while achieving malicious security. We refer the reader to [Section 5.2](#) for more details of the protocol and other subtle issues we faced in our protocol design.

**Special cases.** As we mentioned above, the two-round protocol of Gordon et al. [GLS15] with broadcast and PKI can be transformed into a  $(t + 2)$ -round protocol if the broadcast in the first round is instantiated by a  $(t + 1)$ -round protocol over pairwise-private channels [DS83] and parties only send their messages to  $Q$  in the second round. For  $t = 1$  and 2, we can achieve better than five rounds. For  $t = 1$ , when  $Q$  does not have input, we can design a two-round protocol which crucially relies on the fact that at most one party is corrupted. See [Section 6.2.1](#) for more details.

## 2.2 Overview of Lower Bounds

At a high level, we use the following common approach in our lower-bound arguments. To prove the impossibility of an  $r$ -round protocol, we assume towards a contradiction that an  $r$ -round solitary MPC protocol  $\Pi$  with **god** exists. Next, we analyze a sequence of scenarios which allow us to draw inferences regarding the properties that  $\Pi$  must satisfy. Here, we exploit the guarantees of correctness, privacy and full-security (guaranteed output delivery). The strategic design of the scenarios building on these inferences lets us arrive at the final contradiction. The crux of these arguments lies in the adversarial strategies, which exploit either the absence of broadcast or PKI, depending on the relevant network model.

**With broadcast and no PKI.** For our three-round lower bound with a broadcast channel and no PKI setup, in accordance with the above, we assume there exists a two-round protocol  $\Pi$  with **god** computing a 3-party solitary functionality amongst parties  $P_1, P_2$ , and  $Q$  (output receiving party). The first two scenarios involving a malicious  $P_2$  and passive  $Q$  respectively allow us to infer the following property of  $\Pi$  –  $\Pi$  must be such that even if  $P_2$  does not communicate privately to  $Q$  in Round 1 and aborts in Round 2,  $Q$  must still be able to compute the output on  $x_2$  i.e. the input with respect to which it interacted with  $P_1$  in Round 1. This can be inferred by the security guarantee of **god**. Intuitively, this implies that  $Q$  relies on the following messages to carry information about  $x_2$  required for output computation (i)  $P_1$ 's broadcast message in Round 2 and (ii)  $P_2$ 's broadcast message in Round 1. However, we note that, both of these are also available to  $P_1$  at the end of Round 1 itself. This leads us to a final scenario, in that a passive  $P_1$  can compute the residual function  $f(\widetilde{x}_1, x_2, \widetilde{x}_3)$  for more than one choices of  $(\widetilde{x}_1, \widetilde{x}_3)$ , while the input of honest  $P_2$  remains fixed – which is the final contradiction. Notably, this argument does not work in the presence of a PKI, which allows  $Q$  to have some secret, such as a secret-key, under which  $P_2$ 's messages can be encrypted – this disables  $P_1$  to recover the same information as  $Q$  after Round 1.

**Necessity of broadcast in Round 1.** To show the necessity of broadcast in Round 1 in a three-round solitary MPC protocol with **god** (with broadcast and no PKI), we assume there exists a three-round protocol  $\Pi$  with **god** computing a 3-party solitary functionality among parties  $\{P_1, P_2, Q\}$  which uses the broadcast channel only in Round 2 and Round 3 (and uses pairwise-private channels

in all rounds). We first consider a scenario with a malicious  $P_2$ , who only behaves honestly to  $P_1$  and pretends to have received a maliciously computed message from  $Q$  in Round 1. In addition,  $P_2$  aborts in Round 3. We show that an honest  $Q$  in this scenario must obtain  $f(x_1, x_2, x_3)$  as the output, where  $x_1, x_2, x_3$  are the parties' honest inputs. First of all,  $Q$  must learn an output computed on the honest parties' inputs  $x_1$  and  $x_3$  by the **god** property of  $\Pi$ . The output is also w.r.t.  $P_2$ 's honest input  $x_2$  because  $Q$ 's view in this scenario is subsumed by another scenario with a malicious  $Q$ , where  $Q$  only behaves honestly to  $P_1$  and pretends to have received a maliciously computed message from  $P_2$  in Round 1. Since the first-round messages are only sent via pairwise-private channels,  $P_1$  cannot distinguish whether  $P_2$  is malicious (first scenario) or  $Q$  is malicious (second scenario), and  $P_1$ 's view is identically distributed in both scenarios. Comparing the messages received by  $Q$  in the two scenarios, we can conclude  $Q$ 's view in the first scenario is subsumed by its view in the second scenario. Notice that a malicious  $Q$  in the second scenario can only learn an output on the honest parties' input  $x_1$  and  $x_2$ , hence  $Q$  must learn  $f(x_1, x_2, x_3)$  in both scenarios. The key takeaway is that  $P_2$ 's input can be considered as "committed" in its private message to  $P_1$  in Round 1 and broadcast message in Round 2. This allows a semi-honest  $P_1$  to emulate  $Q$ 's view in the first scenario for an arbitrary input  $\widetilde{x}_3$  and locally compute  $f(x_1, x_2, \widetilde{x}_3)$  violating the security of  $\Pi$ . This argument can be extended for the general case  $n/3 \leq t < n/2$ . A more detailed proof is presented in [Section 4.2](#).

**Necessity of broadcast in Round 2.** For our result showing necessity of broadcast in Round 2, we assume there exists a three-round 5-party solitary MPC  $\Pi$  with **god** against 2 corruptions which uses broadcast in only Round 1 and Round 3 (and uses pairwise-private channels in all rounds). The argument involves two crucial observations **(1)**  $\Pi$  is such that if corrupt  $P_1$  participates honestly using input  $x_1$  *only* in the broadcast communication and private communication towards  $\{P_2, P_5 = Q\}$  in Round 1 (and sends no other messages during  $\Pi$ ), then there exists some  $x_1$ , say  $x_1^*$ , such that the output obtained by  $Q$  is *not* computed with respect to  $x_1^*$  with non-negligible probability. Intuitively, if this does not hold and for all  $x_1$  the output is computed with respect to  $x_1$ , then it would mean that  $\Pi$  is such that  $\{P_2, Q\}$  obtain sufficient information to compute on  $x_1$  at the end of Round 1 itself. This would make  $\Pi$  susceptible to residual function attack by  $\{P_2, Q\}$  which violates security. **(2)**  $\Pi$  is such that if corrupt  $\{P_3, P_4\}$  pretend in Round 2 as if they have not received private communication from  $P_1$  in Round 1, still, the output obtained by  $Q$  must be computed on honest  $P_1$ 's input  $x_1$ . This follows from correctness of  $\Pi$ . Next, we design a final scenario building on **(1)** and **(2)** where an adversary corrupting  $\{P_1, Q\}$  obtains multiple outputs, with respect to both input  $x_1' \neq x_1^*$  and  $x_1^*$ ; which gives the final contradiction. Crucially, due to absence of broadcast in Round 2, the adversary is able to keep the honest parties  $\{P_2, P_3, P_4\}$  on different pages with respect to whether  $P_1$  has aborted after Round 1 or not. Specifically, the adversarial strategy in the final scenario exploits the absence of broadcast in Round 2 to ensure the following - (a) view of honest  $\{P_3, P_4\}$  is similar to the scenario in **(1)**, where they do not receive any communication from  $P_1$  except its broadcast communication in Round 1 and (b) view of honest  $P_2$  is similar to the scenario in **(2)**. Here,  $P_2$  receives communication from  $P_1$  in both Round 1 and Round 2; but receives communication from  $\{P_3, P_4\}$  in Round 2 conveying that they did not receive  $P_1$ 's private communication in Round 1 (the Round 2 messages from  $\{P_3, P_4\}$  could potentially convey this information, depending on protocol design). This inconsistency in the views of honest parties enables the adversary to obtain multiple outputs. This completes the description focusing on the core ideas, we defer the more involved technicalities to [Section 4.3](#). This argument

is extended for the general case  $2 \lceil n/5 \rceil \leq t < n/2$ .

**With PKI and no broadcast.** The lower-bound arguments in the setting with a PKI setup and no broadcast tend to be more involved as PKI can be used to allow output obtaining party  $Q$  to have some secret useful for output computation (as elaborated in the overview of 3-round lower bound above). For our four-round general lower bound that holds for  $3 \lceil n/7 \rceil \leq t < n/2$  and  $t \geq 3$ , we assume there exists a three-round protocol  $\Pi$  with  $\text{god}$  computing a special 7-party solitary functionality amongst parties  $P_1, \dots, P_6$  and  $Q$ . We analyze four main scenarios as follows. In Scenarios 1 and 2,  $\{P_1, P_6\}$  are corrupt and  $P_1$  does not communicate directly to anyone throughout. The crucial difference between them is in the communication of  $P_6$  in Round 2 to  $\{P_2, P_3, P_4, P_5\}$ : in Scenario 1,  $P_6$  acts as if he *did not receive* any communication from  $P_1$  in Round 1; in Scenario 2,  $P_6$  pretends to *have received* communication from  $P_1$  in Round 1. We first show that in Scenario 1,  $Q$  must learn an output computed with regard to some  $x_1^*$  that is independent of  $x_1$ , with non-negligible probability. Intuitively, this holds because the communication in Scenario 1 is independent of  $x_1$ . Next, consider this special  $x_1^*$ , we prove via a sequence of hybrids that Scenario 2 must also result in output being computed on  $x_1^*$  with non-negligible probability. This lets us infer a critical property satisfied by  $\Pi$  - if  $\{P_3, P_4, P_5\}$  do not receive any communication *directly* from  $P_1$  throughout  $\Pi$  and only potentially receive information regarding  $P_1$  *indirectly* via  $P_6$  (say  $P_6$  claims to have received authenticated information from  $P_1$  which can be verified by  $\{P_3, P_4, P_5\}$  due to availability of PKI), then  $Q$  obtains an output on some  $x_1^*$  that is independent of  $x_1$ , with non-negligible probability.

Next, we consider an orthogonal scenario (Scenario 3) where  $\{P_3, P_4, P_5\}$  are corrupt and pretend as if they received no information from  $P_1$  directly. Correctness of  $\Pi$  ensures that  $Q$  must obtain output on honest input of  $P_1$  using the messages from  $\{P_1, P_2, P_6\}$ . Roughly speaking, the above observations enable us to partition the parties  $\{P_1, \dots, P_6\}$  into two sets  $\{P_1, P_2, P_6\}$  and  $\{P_3, P_4, P_5\}$ . Combining the above inferences, we design the final scenario where adversary corrupts  $\{P_1, P_2, Q\}$  and participates with  $x_1' \neq x_1^*$ . Here,  $P_1$  behaves honestly only to  $P_6$  (among the honest parties). The communication of corrupt parties is carefully defined so that the following holds: (a) the views of  $\{P_3, P_4, P_5\}$  are identically distributed to their views in Scenario 2, and (b) the views of  $\{P_1, P_2, P_6\}$  are identically distributed to their views in Scenario 3. We then demonstrate that  $Q$  can obtain an output computed on  $x_1^*$  as well as another output computed on  $P_1$ 's honest input  $x_1'$  by using the communication from  $\{P_3, P_4, P_5\}$  and  $\{P_1, P_2, P_6\}$  *selectively*. Several technicalities and intermediate scenarios have been skipped in the above skeleton in the interest of highlighting the key essence of our ideas. We refer to [Section 5.1](#) for the details.

We observe that the above approach inherently demands the presence of 3 or more corruptions. The main bottleneck in extending it to  $t = 2$  arises from the sequence of hybrids between Scenario 1 and 2, which requires the presence of an additional corruption besides  $\{P_1, P_6\}$ . This shows hope for better upper bounds (less than four rounds) for lower corruption thresholds. In this direction, we investigated the cases of  $t = 1$  and  $t = 2$  separately. We showed the necessity of three rounds for  $t = 1$  when  $Q$  has input and for  $t = 2$  (irrespective of whether  $Q$  has input). These lower bounds also employ the common approach outlined above but differ significantly in terms of the associated scenarios. We refer to the respective technical sections for details. Notably, all the lower bounds also extend to arbitrary correlated randomness setup.

## 3 Preliminaries

### 3.1 Notation and Setting

We use  $\lambda$  to denote the security parameter. By  $\text{poly}(\lambda)$  we denote a polynomial function in  $\lambda$ . By  $\text{negl}(\lambda)$  we denote a negligible function, that is, a function  $f$  such that  $f(\lambda) < 1/p(\lambda)$  holds for any polynomial  $p(\cdot)$  and sufficiently large  $\lambda$ . We use  $\llbracket x \rrbracket$  to denote an encryption of  $x$ .

We consider a set of parties  $\{P_1 \dots, P_n\}$ . Each party is modelled as a probabilistic polynomial-time (PPT) Turing machine. We assume that there exists a PPT adversary who can corrupt up to  $t$  parties where  $n/3 \leq t < n/2$ . We assume throughout that the parties are connected by pairwise-secure and authentic channels and have access to a common reference string (CRS). Additional setup or network assumption is explicitly mentioned in the respective sections.

### 3.2 Security Model

We prove the security of our protocols based on the standard real/ideal world paradigm. Essentially, the security of a protocol is analyzed by comparing what an adversary can do in the real execution of the protocol to what it can do in an ideal execution, that is considered secure by definition (in the presence of an incorruptible trusted party). In an ideal execution, each party sends its input to the trusted party over a perfectly secure channel, the trusted party computes the function based on these inputs and sends to each party its respective output. Informally, a protocol is secure if whatever an adversary can do in the real protocol (where no trusted party exists) can be done in the above described ideal computation. We formalize the model below with text taken verbatim from [CL14] and refer to [CL14] for further details.

**Execution in the Real World.** Throughout a real execution, all the honest parties follow the instructions of the prescribed protocol, whereas the corrupted parties receive their instructions from the adversary. Then, at the conclusion of the execution, the honest parties output their prescribed output from the protocol, the corrupted parties output nothing and the adversary outputs an (arbitrary) function of its view of the computation (which contains the views of all the corrupted parties). Without loss of generality, we assume that the adversary always outputs its view (and not some function of it).

**Definition 3.1.** (*Real-model execution*). Let  $f$  be an  $n$ -party functionality, let  $\pi$  be a multiparty protocol for computing  $f$  and let  $\lambda$  be the security parameter. Let  $\mathcal{I} \subseteq [n]$  denotes the set of indices of the parties corrupted by  $\mathcal{A}$ . Then, the joint execution of  $\pi$  under  $(\mathcal{A}, \mathcal{I})$  in the real model, on input vector  $\bar{x} = (x_1, \dots, x_n)$ , auxiliary input  $z$  to  $\mathcal{A}$  and security parameter  $\lambda$ , denoted  $\text{REAL}_{\pi, \mathcal{I}, \mathcal{A}(z)}(\bar{x}, \lambda)$  is denoted as the output vector of  $P_1, \dots, P_n$  and  $\mathcal{A}$  resulting from the protocol interaction, where for every  $i \in \mathcal{I}$ , party  $P_i$  computes its messages according to  $\mathcal{A}$ , and for every  $j \notin \mathcal{I}$ , party  $P_j$  computes its messages according to  $\pi$ .

**Execution in the Ideal World.** For full security a.k.a security with *guaranteed output delivery*, an ideal execution proceeds as follows:

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

- *Trusted party answers the parties:* If  $x'_i$  is outside of the domain for  $P_i$  or  $P_i$  sends no input, the trusted party sets  $x'_i$  to be a predetermined default value  $\hat{x}_i$ . Next, the trusted party computes  $f(x'_1, \dots, x'_n) = (y_1, \dots, y_n)$  and sends  $y_i$  to party  $P_i$  for every  $i$ .
- *Outputs:* Honest parties always output the message received from the trusted party and the corrupted parties output nothing. The adversary outputs an arbitrary function of the initial  $\{x_i\}_{i \in \mathcal{I}}$  and the messages received by the corrupted parties from the trusted party  $\{y_i\}_{i \in \mathcal{I}}$ , where  $\mathcal{I}$  denotes the set of indices of the corrupted parties.

**Definition 3.2.** (*Ideal-model execution with guaranteed output delivery*). Let  $f : (\{0, 1\}^*)^n \rightarrow (\{0, 1\}^*)^n$  be an  $n$ -party functionality where  $f = (f_1, \dots, f_n)$ . Let  $\lambda$  be the security parameter and  $\mathcal{I} \subseteq [n]$  denotes the set of indices of the corrupted parties. Then, the joint execution of  $f$  under  $(\text{Sim}, \mathcal{I})$  in the ideal model, on input vector  $\bar{x} = (x_1, \dots, x_n)$ , auxiliary input  $z$  to  $\text{Sim}$  and security parameter  $\lambda$ , denoted  $\text{IDEAL}_{f, \mathcal{I}, \text{Sim}(z)}(\bar{x}, \lambda)$  is denoted as the output vector of  $P_1, \dots, P_n$  and  $\text{Sim}$  resulting from the above described ideal process.

**Security of Protocols.** The security of protocols is formulated by saying that adversaries in the ideal model are able to simulate adversaries in an execution of a protocol in the real model.

**Definition 3.3.** Let  $f : (\{0, 1\}^*)^n \rightarrow (\{0, 1\}^*)^n$  be an  $n$ -party functionality and let  $\pi$  be a protocol computing  $f$ . We say that protocol  $\pi$  ***t*-securely computes  $f$  with guaranteed output delivery (god)** if for every non-uniform polynomial-time adversary  $\mathcal{A}$  for the real model, there exists a non-uniform probabilistic (expected) polynomial-time adversary  $\text{Sim}$  for the ideal model, such that for every  $\mathcal{I} \subseteq [n]$  with  $|\mathcal{I}| \leq t$ , the following two distributions are computationally indistinguishable:

$$\left\{ \text{REAL}_{\pi, \mathcal{I}, \mathcal{A}(z)}(\bar{x}, \lambda) \right\}_{(\bar{x}, z) \in (\{0, 1\}^*)^{n+1}, \kappa \in \mathbb{N}} \quad \text{and} \quad \left\{ \text{IDEAL}_{f, \mathcal{I}, \text{Sim}(z)}(\bar{x}, \lambda) \right\}_{(\bar{x}, z) \in (\{0, 1\}^*)^{n+1}, \kappa \in \mathbb{N}}$$

**Solitary Output.** In this work, we consider the setting where the output is delivered to only one party  $Q = P_n$ . That is, we consider functions  $f$  where  $f(x_1, \dots, x_n) = (\perp, \dots, \perp, y_n)$ .

### 3.3 Cryptographic Primitives

#### 3.3.1 Digital Signatures

A digital signature scheme consists of the following three algorithms ( $\text{Gen}, \text{Sign}, \text{Verify}$ ).

- $(\text{skey}, \text{vkey}) \leftarrow \text{Gen}(1^\lambda)$ . A randomized algorithm that takes the security parameter  $\lambda$  as input, and generates a verification-key  $\text{vkey}$  and a signing key  $\text{skey}$ .
- $\sigma \leftarrow \text{Sign}(\text{skey}, m)$ . A randomized algorithm that takes a message  $m$  and signing key  $\text{skey}$  as input and outputs a signature  $\sigma$ .
- $0/1 \leftarrow \text{Verify}(\text{vkey}, (m, \sigma))$ . A deterministic algorithm that takes a verification key  $\text{vkey}$  and a candidate message-signature pair  $(m, \sigma)$  as input, and outputs 1 for a valid signature and 0 otherwise.

The following correctness and security properties should be satisfied:

- **Correctness.** For all  $\lambda \in \mathbb{N}$ , all  $(\text{vkey}, \text{skey}) \leftarrow \text{Gen}(1^\lambda)$ , any message  $m$ ,  $\text{Verify}(\text{vkey}, m, \text{Sign}(\text{skey}, m)) = 1$ .

- **Unforgeability.** A signature scheme is unforgeable if for any PPT adversary  $\mathcal{A}$ , the following game outputs 1 with negligible probability (in security parameter).
  - *Initialize.* Run  $(vkey, skey) \leftarrow \text{Gen}(1^\lambda)$ . Give  $vkey$  to  $\mathcal{A}$ . Initiate a list  $\mathcal{L} = \emptyset$ .
  - *Signing queries.* On query  $m$ , return  $\sigma \leftarrow \text{Sign}(skey, m)$ . Run this step as many times as  $\mathcal{A}$  desires. Then, insert  $m$  into the list  $\mathcal{L}$ .
  - *Output.* Receive output  $(m^*, \sigma^*)$  from  $\mathcal{A}$ . Return 1 if and only if  $\text{Verify}(vkey, (m^*, \sigma^*)) = 1$  and  $m^* \notin \mathcal{L}$ , and 0 otherwise.

### 3.3.2 Simulation-Extractible NIZK

A simulation-extractible non-interactive zero-knowledge (NIZK) argument for an NP language  $L$  with relation  $R$  consists of the following randomized algorithms (NIZK.Setup, NIZK.Prove, NIZK.Verify):

- $\text{crs} \leftarrow \text{NIZK.Setup}(1^\lambda)$  : Given security parameter  $\lambda$ , it outputs a common reference string  $\text{crs}$ .
- $\pi \leftarrow \text{NIZK.Prove}(\text{crs}, \text{st}, \text{wit})$  : Given  $\text{crs}$ , a statement  $x$  and witness  $w$ , outputs a proof  $\pi$ .
- $0/1 \leftarrow \text{NIZK.Verify}(\text{crs}, \text{st}, \pi)$  : Given  $\text{crs}$ , a statement  $x$  and proof  $\pi$ , it outputs one bit.

It should satisfy the following correctness and security properties.

- **Completeness.** For every security parameter  $\lambda \in \mathbb{N}$ , and any  $(\text{st}, \text{wit}) \in R$ ,

$$\Pr[\text{NIZK.Verify}(\text{crs}, \pi, \text{st}) = 1 : \text{crs} \leftarrow \text{NIZK.Setup}(1^\lambda), \\ \pi \leftarrow \text{NIZK.Prove}(\text{crs}, \text{st}, \text{wit})] \geq 1 - \text{negl}(\lambda)$$

where the probability is over the randomness of the three algorithms.

- **Zero Knowledge.** For any malicious verifier  $\mathcal{A}_V$ , there exists a PPT simulator  $(\text{NIZK.Sim.Setup}, \text{NIZK.Sim.Prove})$  such that for all  $(\text{st}, \text{wit}) \in R$ :

$$(\text{crs}, \pi) \stackrel{c}{\approx} (\text{simcrs}, \pi^*)$$

where  $\text{crs} \leftarrow \text{NIZK.Setup}(1^\lambda)$ ,  $\pi \leftarrow \text{NIZK.Prove}(\text{crs}, \text{st}, \text{wit})$ ,  $(\text{simcrs}, \text{td}) \leftarrow \text{NIZK.Sim.Setup}(1^\lambda)$ ,  $\pi^* \leftarrow \text{NIZK.Sim.Prove}(\text{td}, \text{st})$  and the probability is over the randomness of all algorithms.

- **Simulation Extractibility.** For any PPT cheating prover  $\mathcal{A}_P$ , there exists a PPT extractor  $\text{NIZK.Sim.Ext}$  such that for all  $\text{st}$ :

$$\Pr[\text{NIZK.Verify}(\text{simcrs}, \pi^*, \text{st}) = 1 \wedge \text{wit} = \text{NIZK.Sim.Ext}(\text{td}, \pi^*, \text{st}) \\ \wedge (\text{st}, \text{wit}) \notin R \wedge (\text{st}, \pi^*) \notin \mathcal{L}] \leq \text{negl}(\lambda)$$

where  $\pi^* = \mathcal{A}_P^{\text{NIZK.Sim.Prove}(\text{td}, \cdot)}(\text{simcrs}, \text{st})$ ,  $(\text{simcrs}, \text{td}) \leftarrow \text{NIZK.Sim.Setup}(1^\lambda)$ ,  $\mathcal{L}$  is the set of  $(\text{st}_i, \pi_i)$  responses output by the oracle  $\text{NIZK.Sim.Prove}(\text{simcrs}, \cdot)$  that  $\mathcal{A}_P$  gets access to and the probability is over the randomness of all algorithms.

**Languages Used.** In our solitary MPC protocols presented in [Section 5.2](#) and [Section 6.2.1](#), we will consider two NP languages  $L_1, L_2$  for the NIZK described below.

- **NP Language  $L_1$ :**  
Statement  $\text{st} = (\llbracket x \rrbracket, \text{pk})$       Witness  $\text{wit} = (x, \rho)$   
 $R_1(\text{st}, \text{wit}) = 1$  iff  $\llbracket x \rrbracket = \text{dTFHE.Enc}(\text{pk}, x; \rho)$ .

- **NP Language  $L_2$ :**

Statement  $\text{st} = (\llbracket x : \text{sk} \rrbracket, \llbracket x \rrbracket, \text{pk}, i)$       Witness  $\text{wit} = (\text{sk}, r)$

$R_2(\text{st}, \text{wit}) = 1$  iff  $\llbracket x : \text{sk} \rrbracket = \text{dTFHE.PartialDec}(\text{sk}, \llbracket x \rrbracket)$  and  $(\text{pk}, \text{sk}) = \text{dTFHE.DistGen}(1^\lambda, 1^d, i; r)$ .

**Imported Theorem 1** ([CCH<sup>+</sup>19, PS19]). *Assuming LWE, there exists a Simulation-Extractible NIZK argument system for all NP languages.*

### 3.3.3 Threshold Fully Homomorphic Encryption

We define a  $t$ -out-of- $n$  decentralized threshold fully homomorphic encryption scheme as in the work of Boneh et al. [BGG<sup>+</sup>18].

**Definition 3.4. (Decentralized Threshold Fully Homomorphic Encryption (dTFHE))**

Let  $\mathcal{P} = \{P_1, \dots, P_n\}$  be a set of parties. A dTFHE scheme is a tuple of PPT algorithms  $\text{dTFHE} = (\text{dTFHE.DistGen}, \text{dTFHE.Enc}, \text{dTFHE.PartialDec}, \text{dTFHE.Eval}, \text{dTFHE.Combine})$  with the following syntax:

- $(\text{pk}_i, \text{sk}_i) \leftarrow \text{dTFHE.DistGen}(1^\lambda, 1^d, i; r_i)$ : On input the security parameter  $\lambda$ , a depth bound  $d$ , party index  $i$  and randomness  $r_i$ , the distributed setup outputs a public-secret key pair  $(\text{pk}_i, \text{sk}_i)$  for party  $P_i$ . We denote the public key of the scheme as  $\text{pk} = (\text{pk}_1 \parallel \dots \parallel \text{pk}_n)$ .
- $\llbracket m \rrbracket \leftarrow \text{dTFHE.Enc}(\text{pk}, m)$ : On input a public key  $\text{pk}$ , and a plaintext  $m$  in the message space  $\mathcal{M}$ , it outputs a ciphertext  $\llbracket m \rrbracket$ .
- $\llbracket y \rrbracket \leftarrow \text{dTFHE.Eval}(\text{pk}, \text{C}, \llbracket m_1 \rrbracket, \dots, \llbracket m_k \rrbracket)$ : On input a public key  $\text{pk}$ , a circuit  $\text{C}$  of depth at most  $d$  that takes  $k$  inputs each from the message space and outputs one value in the message space, and a set of ciphertexts  $\llbracket m_1 \rrbracket, \dots, \llbracket m_k \rrbracket$  where  $k = \text{poly}(\lambda)$ , the evaluation algorithm outputs a ciphertext  $\llbracket y \rrbracket$ .
- $\llbracket m : \text{sk}_i \rrbracket \leftarrow \text{dTFHE.PartialDec}(\text{sk}_i, \llbracket m \rrbracket)$ : On input a secret key share  $\text{sk}_i$  and a ciphertext  $\llbracket m \rrbracket$ , it outputs a partial decryption  $\llbracket m : \text{sk}_i \rrbracket$ .
- $m/\perp \leftarrow \text{dTFHE.Combine}(\text{pk}, \{\llbracket m : \text{sk}_i \rrbracket\}_{i \in S})$ : On input a public key  $\text{pk}$  and a set of partial decryptions  $\{\llbracket m : \text{sk}_i \rrbracket\}_{i \in S}$  where  $S \subseteq [n]$ , the combination algorithm either outputs a plaintext  $m$  or the symbol  $\perp$ .

As in a standard homomorphic encryption scheme, we require that a dTFHE scheme satisfies compactness, correctness and security. We discuss these properties below.

**Compactness.** A dTFHE scheme is said to be compact if there exists polynomials  $\text{poly}_1(\cdot)$  and  $\text{poly}_2(\cdot)$  for all  $\lambda$ , all message spaces  $\mathcal{M}$  with size of each message being  $\text{poly}_3(\lambda)$ , all  $k = \text{poly}(\lambda)$ , depth bound  $d$ , circuit  $\text{C} : \{0, 1\}^{k \cdot \text{poly}_3(\lambda)} \rightarrow \{0, 1\}^\lambda$  of depth at most  $d$  and  $m_i \in \mathcal{M}$  for  $i \in [k]$ , the following condition holds. Let  $(\text{pk}_j, \text{sk}_j) \leftarrow \text{dTFHE.DistGen}(1^\lambda, 1^d, j)$  for all  $j \in [n]$ ,  $\text{pk} = (\text{pk}_1 \parallel \dots \parallel \text{pk}_n)$ ; let  $\llbracket m_i \rrbracket \leftarrow \text{dTFHE.Enc}(\text{pk}, m_i)$  for all  $i \in [k]$ ; compute  $\llbracket y \rrbracket \leftarrow \text{dTFHE.Eval}(\text{pk}, \text{C}, \llbracket m_1 \rrbracket, \dots, \llbracket m_k \rrbracket)$  and  $\llbracket y : \text{sk}_j \rrbracket \leftarrow \text{dTFHE.PartialDec}(\text{sk}_j, \llbracket y \rrbracket)$  for  $j \in [n]$ , then it holds that  $|\llbracket y \rrbracket| \leq \text{poly}_1(\lambda, n, d)$  and  $|\llbracket y : \text{sk}_j \rrbracket| \leq \text{poly}_2(\lambda, n, d)$ .

**Evaluation Correctness.** Informally, a dTFHE scheme is said to be correct if recombining partial decryptions of a ciphertext output by the evaluation algorithm returns the correct evaluation of the corresponding circuit on the underlying plaintexts. Formally, We say that a dTFHE scheme satisfies evaluation correctness if for all  $\lambda$ , all message spaces  $\mathcal{M}$  with size of each message being  $\text{poly}_3(\lambda)$ , all  $k = \text{poly}(\lambda)$ , circuit  $C : \{0, 1\}^{k \cdot \text{poly}_3(\lambda)} \rightarrow \{0, 1\}^\lambda$  of depth at most  $d$  and  $m_i \in \mathcal{M}$  for  $i \in [k]$ , the following condition holds. Let  $(\text{pk}_j, \text{sk}_j) \leftarrow \text{dTFHE.DistGen}(1^\lambda, 1^d, j)$  for all  $j \in [n]$ ,  $\text{pk} = (\text{pk}_1 \parallel \dots \parallel \text{pk}_n)$ ; let  $\llbracket m_i \rrbracket \leftarrow \text{dTFHE.Enc}(\text{pk}, m_i)$  for all  $i \in [k]$  and  $\llbracket y \rrbracket \leftarrow \text{dTFHE.Eval}(\text{pk}, C, \llbracket m_1 \rrbracket, \dots, \llbracket m_k \rrbracket)$ , for any  $S \subseteq [n]$ ,  $|S| = t$ ,

$$\Pr [\text{dTFHE.Combine}(\text{pk}, \{\text{dTFHE.PartialDec}(\text{sk}_j, \llbracket y \rrbracket)\}_{j \in S}) = C(m_1, \dots, m_k)] \geq 1 - \text{negl}(\lambda).$$

**Semantic Security.** Informally, a dTFHE scheme is said to provide semantic security if a PPT adversary cannot efficiently distinguish between encryptions of arbitrarily chosen plaintext messages  $m_0$  and  $m_1$ , even given the secret key shares corresponding to a subset  $S$  of the parties for any set  $S$  of size at most  $(t - 1)$ . Formally, a dTFHE scheme satisfies semantic security if for all  $\lambda$ , all depth bound  $d$ , message space  $\mathcal{M}$ , for any PPT adversary  $\mathcal{A}$ , the following experiment  $\text{Expt}_{\text{dTFHE,sem}}(1^\lambda)$  outputs 1 with negligible probability:

$\text{Expt}_{\text{dTFHE,sem}}(1^\lambda, 1^d)$ :

1. On input the security parameter  $1^\lambda$ , circuit depth  $1^d$ , the message space  $\mathcal{M}$  and the number of parties  $n$ , the adversary  $\mathcal{A}$  outputs a set  $S$  of size at most  $(t - 1)$  and two messages  $m_0, m_1 \in \mathcal{M}$ .
2. The challenger generates  $(\text{pk}_j, \text{sk}_j) \leftarrow \text{dTFHE.DistGen}(1^\lambda, 1^d, j)$  for all  $j \in [n]$ , sets  $\text{pk} = (\text{pk}_1 \parallel \dots \parallel \text{pk}_n)$  and provides  $(\text{pk}, \{\text{sk}_i\}_{i \in S})$  along with  $\text{dTFHE.Enc}(\text{pk}, m_b)$  to  $\mathcal{A}$  where  $b$  is picked uniformly at random.
3.  $\mathcal{A}$  outputs a guess  $b'$ . The experiment outputs 1 if  $b = b'$ .

**Simulation Security.** Informally, a dTFHE scheme is said to provide simulation security if there exists an efficient algorithm  $\text{dTFHE.Sim}$  that takes as input a circuit  $C$ , a set of ciphertexts  $\llbracket m_1 \rrbracket, \dots, \llbracket m_k \rrbracket$ , the output of  $C$  on the corresponding plaintexts, and outputs a set of partial decryptions corresponding to some subset of parties, such that its output is computationally indistinguishable from the output of a real algorithm that homomorphically evaluates the circuit  $C$  on the ciphertexts  $\llbracket m_1 \rrbracket, \dots, \llbracket m_k \rrbracket$  and outputs partial decryptions using the corresponding secret key shares for the same subset of parties. In particular, the computational indistinguishability holds even when a PPT adversary is given the secret key shares corresponding to a subset  $S$  of the parties, so long as  $\text{dTFHE.Sim}$  also gets the secret key shares corresponding to the parties in  $S$ .

Formally, a dTFHE scheme satisfies simulation security if for all  $\lambda$ , all depth bound  $d$ , message space  $\mathcal{M}$ , for any PPT adversary  $\mathcal{A}$ , there exists a simulator  $\text{dTFHE.Sim}$  such that the following two experiments  $\text{Expt}_{\text{dTFHE,Real}}(1^\lambda)$  and  $\text{Expt}_{\text{dTFHE,Ideal}}(1^\lambda)$  are computationally indistinguishable.

$\text{Expt}_{\text{dTFHE,Real}}(1^\lambda, 1^d)$ :

1. On input the security parameter  $1^\lambda$ , circuit depth  $d$ , the message space  $\mathcal{M}$  and the number of parties  $n$ , the adversary  $\mathcal{A}$  outputs a set  $S$  of size at most  $(t - 1)$  and a set of messages  $m_1, \dots, m_k$  for  $k = \text{poly}(\lambda)$ .

2. The challenger generates  $(\mathbf{pk}_j, \mathbf{sk}_j) \leftarrow \text{dTFHE.DistGen}(1^\lambda, 1^d, j)$  for all  $j \in [n]$ , sets  $\mathbf{pk} = (\mathbf{pk}_1 \parallel \dots \parallel \mathbf{pk}_n)$  and provides  $(\mathbf{pk}, \{\mathbf{sk}_i\}_{i \in S})$  along with  $\llbracket m_i \rrbracket \leftarrow \text{dTFHE.Enc}(\mathbf{pk}, m_i)$  for each  $i \in [k]$  to  $\mathcal{A}$ .
3.  $\mathcal{A}$  issues a query with a circuit  $C$ . The challenger first computes  $\llbracket y \rrbracket \leftarrow \text{dTFHE.Eval}(\mathbf{pk}, C, \llbracket m_1 \rrbracket, \dots, \llbracket m_k \rrbracket)$ . Then, it outputs  $\{\text{dTFHE.PartialDec}(\mathbf{sk}_i, \llbracket y \rrbracket)\}_{i \notin S}$  to  $\mathcal{A}$ .
4. At the end of the experiment,  $\mathcal{A}$  outputs a distinguishing bit  $b$ .

$\text{Expt}_{\text{dTFHE, Ideal}}(1^\lambda, 1^d)$ :

1. On input the security parameter  $1^\lambda$ , circuit depth  $d$ , the message space  $\mathcal{M}$  and the number of parties  $n$ , the adversary  $\mathcal{A}$  outputs a set  $S$  of size at most  $(t - 1)$  and a set of messages  $m_1, \dots, m_k$  for  $k = \text{poly}(\lambda)$ .
2. The challenger generates  $(\mathbf{pk}_j, \mathbf{sk}_j) \leftarrow \text{dTFHE.DistGen}(1^\lambda, 1^d, j)$  for all  $j \in [n]$ , sets  $\mathbf{pk} = (\mathbf{pk}_1 \parallel \dots \parallel \mathbf{pk}_n)$  and provides  $(\mathbf{pk}, \{\mathbf{sk}_i\}_{i \in S})$  along with  $\llbracket m_i \rrbracket \leftarrow \text{dTFHE.Enc}(\mathbf{pk}, m_i)$  for each  $i \in [k]$  to  $\mathcal{A}$ .
3.  $\mathcal{A}$  issues a query with a circuit  $C$ . The challenger outputs  $\text{dTFHE.Sim}(C, C(m_1, \dots, m_k), \llbracket m_1 \rrbracket, \dots, \llbracket m_k \rrbracket, \{\mathbf{sk}_i\}_{i \in S})$  to  $\mathcal{A}$ .
4. At the end of the experiment,  $\mathcal{A}$  outputs a distinguishing bit  $b$ .

**Imported Theorem 2** ([[BGG<sup>+</sup>18](#)]). *Assuming  $\text{LWE}$ , there exists a  $\text{TFHE}$  scheme for every  $t$ -out-of- $n$  threshold access structure.*

## 4 With Broadcast and No PKI

In this section, we assume a network setting where the parties have access to a broadcast channel in addition to pairwise-private channels. In terms of setup, we assume that all parties have access to a common reference string (CRS). First, we present a new lower bound of *three* rounds for solitary MPC with `god` in [Section 4.1](#). Then we study whether it is possible to use fewer rounds of broadcast and show in [Section 4.2](#) and [Section 4.3](#) that broadcast is necessary in both the first and second rounds. Finally, we show that the above negative results are tight by demonstrating the existing results of [[GLS15](#), [ACGJ18](#), [BJMS20](#), [PR18](#)] in [Section 4.4](#).

### 4.1 Necessity of Three Rounds

We show that it is impossible to design a two-round solitary MPC with `god` in the honest majority setting (in particular,  $n/3 \leq t < n/2$ ), assuming the presence of pairwise-private channels and a broadcast channel. Our result holds in the presence of any common public setup such as CRS, even against non-rushing adversaries and irrespective of whether the output-obtaining party  $Q$  provides an input or not.

Before presenting our proof, we first analyze whether the existing lower bounds (three rounds) for standard MPC with `god` in the presence of an honest majority [[GIKR02](#), [GLS15](#), [PR18](#)] hold for solitary functionalities. Among them, [[GIKR02](#), [GLS15](#)] assumes the same network setting as ours while [[GLS15](#)] assumes that the pairwise channels are non-private. For the sake of completeness, we briefly describe below why each of their proof arguments does not hold for solitary MPC. First, we observe that the arguments of [[GLS15](#), [PR18](#)] exploit fairness (implied by `god`) in the following

manner. Their proofs proceed via adversarial strategies where one party gets the output and subsequently draw inferences based on the property that another party must have also obtained the output in that case (due to fairness). Such arguments clearly break down in the context of solitary MPC. Regarding the lower bound of [GIKR02] (which is shown with respect to functions of simultaneous broadcast, XOR and AND of two input bits), we note that it doesn't hold for solitary MPC for the following reason. Their argument which holds for  $t \geq 2$  proceeds via a strategy where the adversary corrupts an input providing party  $P_2$  and another carefully chosen party  $P_j$  ( $j > 2$ ) who should receive the output. The identity of  $P_j$  is determined by identifying the pair of consecutive hybrids which has a non-negligible difference among a sequence of hybrids corresponding to a specific distribution of outputs. This breaks down in case of solitary MPC, as  $Q$  is the only output receiving party and in their lower bound argument, the choice of  $P_j$  may not necessarily result in  $Q$  always. Therefore, existing lower bounds leave the question open regarding the existence of two-round solitary MPC with `god`. We answer this question in the negative and show that three rounds continue to remain the lower bound for `god` even if only a single party is supposed to obtain the output. Notably, our lower bound holds even for a *non-rushing* adversary. We state the formal theorem below.

**Theorem 4.1.** *Assume parties have access to CRS, pairwise-private channels and a broadcast channel. Let  $n$  and  $t$  be positive integers such that  $n \geq 3$  and  $n/3 \leq t < n/2$ . Then, there exists a solitary functionality  $f$  such that no two-round  $n$ -party MPC protocol tolerating  $t$  corruptions can compute  $f$  with `god`, even when the adversary is assumed to be non-rushing.*

*Proof.* For simplicity, we present the argument for the setting  $n = 3$  and  $t = 1$  below and elaborate on how to extend the proof to  $n/3 \leq t < n/2$  later. For the sake of contradiction, suppose there exists a two-round 3-party solitary MPC with `god`, say  $\Pi$  which computes a solitary function  $f(x_1, x_2, x_3)$  among  $\{P_1, P_2, P_3\}$  where  $Q = P_3$  denotes the output receiving party. For simplicity, let  $f$  be defined as  $f(x_1 = (m_0, m_1), x_2 = b, x_3 = \perp) := m_b$ , where  $x_3 = \perp$  denotes that  $Q$  has no input;  $(m_0, m_1) \in \{0, 1\}^\lambda$  denote a pair of strings and  $b \in \{0, 1\}$  denotes a single bit. Note that at most the adversary corrupts at most one party.

We consider three different scenarios of the execution of  $\Pi$ . For simplicity, we assume the following about the structure of  $\Pi$ : **(a)** Round 2 involves only broadcast messages while Round 1 involves messages sent via both pairwise-private and broadcast channels. This holds without loss of generality since the parties can perform pairwise-private communication by exchanging random pads in the first round and then using these random pads to unmask later broadcasts [GIKR01]. **(b)** In Round 1, each pair of parties communicate via their pairwise-private channels (any protocol where a pair of parties does not communicate privately in Round 1 can be transformed to one where dummy messages are exchanged between them). **(c)** Round 2 does not involve any outgoing communication from  $Q$  (as  $Q$  is the only party to receive the output at the end of Round 2).

Next, we define some useful notation: Let  $\text{pc}_{i \rightarrow j}$  denote the pairwise-private communication from  $P_i$  to  $P_j$  in Round 1 and  $\text{b}_{i \rightarrow}^r$  denote the message broadcast by  $P_i$  in round  $r$ , where  $r \in [2], \{i, j\} \in [3]$ . These messages may be a function of the `crs` as per protocol specifications. Let  $\text{View}_i$  denotes the view of party  $P_i$  which consists of `crs`, its input  $x_i$ , randomness  $r_i$  and all incoming messages.

Following is a description of the scenarios. In each of these scenarios, we assume that the adversary uses the honest input on behalf of the corrupt parties and its malicious behaviour is limited to dropping some of the messages supposed to be sent by the corrupt party. The views of the parties for all the scenarios are shown in [Table 3](#).

**Scenario 1:** The adversary actively corrupts  $P_2$  who behaves honestly in Round 1 towards  $P_1$  but doesn't communicate privately to  $Q$  in Round 1. In more detail,  $P_1$  sends messages  $\text{pc}_{2 \rightarrow 1}, \text{b}_{2 \rightarrow}^1$  according to the protocol specification but drops the message  $\text{pc}_{2 \rightarrow 3}$ . In Round 2,  $P_2$  aborts.

**Scenario 2:** The adversary passively corrupts  $Q$  who behaves honestly throughout and learns output  $f(x_1, x_2, x_3)$ . Additionally,  $Q$  locally re-computes the output by emulating Scenario 1, namely when  $P_2$  does not communicate privately to  $Q$  in Round 1 and aborts in Round 2. Specifically,  $Q$  can locally emulate this by discarding  $\text{pc}_{2 \rightarrow 3}$  (private communication from  $P_2$  to  $Q$  in Round 1) and  $\text{b}_{2 \rightarrow}^2$  (broadcast communication from  $P_2$  in Round 2).

**Scenario 3:** The adversary corrupts  $P_1$  passively who behaves honestly throughout.  $P_1$  also does the following local computation: Locally emulate the view of  $Q$  as per Scenario 1 (from which the output can be derived) for various choices of inputs of  $\{P_1, P_3\}$  while the input of  $P_2$  i.e.  $x_2$  remains fixed. In more detail,  $P_1$  does the following - Let  $(\text{pc}_{2 \rightarrow 1}, \text{b}_{2 \rightarrow}^1)$  be fixed to what was received by  $P_1$  in the execution. Choose various combinations of inputs and randomness on behalf of  $P_1$  and  $P_3$ . Consider a particular combination, say  $\{(\widetilde{x}_1, \widetilde{r}_1), (\widetilde{x}_3, \widetilde{r}_3)\}$ . Use it to locally compute  $\widetilde{\text{b}}_{1 \rightarrow}^1, \widetilde{\text{b}}_{3 \rightarrow}^1, \widetilde{\text{pc}}_{1 \rightarrow 3}, \widetilde{\text{pc}}_{3 \rightarrow 1}$ . Next, locally compute  $\widetilde{\text{b}}_{1 \rightarrow}^2$  using the Round 1 emulated messages which results in the complete view  $\widetilde{\text{View}}_3$  of  $Q$  analogous to Scenario 1, where  $\widetilde{\text{View}}_3 = \{\text{crs}, \widetilde{x}_3, \widetilde{r}_3, \widetilde{\text{b}}_{1 \rightarrow}^1, \widetilde{\text{b}}_{2 \rightarrow}^1, \widetilde{\text{pc}}_{1 \rightarrow 3}, \widetilde{\text{b}}_{1 \rightarrow}^2\}$  corresponds to the inputs  $(\widetilde{x}_1, x_2, \widetilde{x}_3)$ .

	Scenario 1			Scenario 2 & 3		
	View <sub>1</sub>	View <sub>2</sub>	View <sub>3</sub>	View <sub>1</sub>	View <sub>2</sub>	View <sub>3</sub>
Initial Input	$(x_1, r_1, \text{crs})$	$(x_2, r_2, \text{crs})$	$(x_3, r_3, \text{crs})$	$(x_1, r_1, \text{crs})$	$(x_2, r_2, \text{crs})$	$(x_3, r_3, \text{crs})$
Round 1	$\text{pc}_{2 \rightarrow 1}, \text{pc}_{3 \rightarrow 1}$ $\text{b}_{2 \rightarrow}^1, \text{b}_{3 \rightarrow}^1$	$\text{pc}_{1 \rightarrow 2}, \text{pc}_{3 \rightarrow 2},$ $\text{b}_{1 \rightarrow}^1, \text{b}_{3 \rightarrow}^1$	$\text{pc}_{1 \rightarrow 3}, \text{pc}_{3 \rightarrow 1},$ $\text{b}_{1 \rightarrow}^1, \text{b}_{2 \rightarrow}^1$	$\text{pc}_{2 \rightarrow 1}, \text{pc}_{3 \rightarrow 1},$ $\text{b}_{2 \rightarrow}^1, \text{b}_{3 \rightarrow}^1$	$\text{pc}_{1 \rightarrow 2}, \text{pc}_{3 \rightarrow 2},$ $\text{b}_{1 \rightarrow}^1, \text{b}_{3 \rightarrow}^1$	$\text{pc}_{1 \rightarrow 3}, \text{pc}_{2 \rightarrow 3},$ $\text{b}_{1 \rightarrow}^1, \text{b}_{2 \rightarrow}^1$
Round 2	–	$\text{b}_{1 \rightarrow}^2$	$\text{b}_{1 \rightarrow}^2$	$\text{b}_{2 \rightarrow}^2$	$\text{b}_{1 \rightarrow}^2$	$\text{b}_{1 \rightarrow}^2, \text{b}_{2 \rightarrow}^2$

Table 3: Views of  $P_1, P_2, P_3$  in Scenarios 1 – 3.

The proof skeleton is as follows. First, we claim that if Scenario 1 occurs, then  $Q$  must obtain  $f(x_1, x_2, x_3)$  with overwhelming probability. If not, then  $\Pi$  is vulnerable to a potential attack by semi-honest  $Q$  (which enables  $Q$  to learn information that he is not supposed to learn) that violates security. Intuitively, this inference captures  $Q$ 's reliance on  $P_1$ 's messages in Round 2 and  $P_2$ 's broadcast in Round 1 to carry information about  $x_2$  required for output computation. Note that this information is available to  $P_1$  at the end of Round 1 itself. Building on this intuition, we show that  $\Pi$  is such that an adversary corrupting  $P_1$  passively can compute  $f(\widetilde{x}_1, x_2, \widetilde{x}_3)$  for any choice of  $(\widetilde{x}_1, \widetilde{x}_3)$ , which is the final contradiction. We now prove a sequence of lemmas to complete our proof.

**Lemma 4.2.**  *$\Pi$  must be such that if Scenario 1 occurs, then  $Q$  outputs  $f(x_1, x_2, x_3)$  with all but negligible probability.*

*Proof.* In Scenario 1, it follows from the god property of  $\Pi$  that  $Q$  obtains an output with all but negligible probability. Since  $P_1$  and  $Q$  are honest, the output must be computed on  $x_1 = (m_0, m_1)$

by the correctness of  $\Pi$ . Therefore,  $Q$ 's output in Scenario 1 is  $f(x_1, x'_2, x_3) = m_{b'}$  for some  $x'_2 = b'$ . Note that the view locally emulated by a semi-honest  $Q$  in Scenario 2 is distributed identically to its view in Scenario 1. We can thus infer that if Scenario 2 occurs, then  $\Pi$  allows a semi-honest  $Q$  to learn  $f(x_1, x'_2, x_3) = m_{b'}$ . By the security of  $\Pi$ ,  $Q$  is not allowed to learn anything beyond the correct output in Scenario 2 (i.e.,  $f(x_1, x_2, x_3) = m_b$  for  $x_2 = b$  as all behave honestly in Scenario 2) as per the ideal functionality. We can thus conclude that if Scenario 1 occurs, then  $b' = b$  and  $Q$  outputs  $y = f(x_1, x_2, x_3) = m_b$  with all but negligible probability.  $\square$

**Lemma 4.3.**  $\Pi$  is such that a semi-honest  $P_1$  can compute the residual function  $f(\widetilde{x}_1, x_2, \widetilde{x}_3)$  (for different choices of  $(\widetilde{x}_1, \widetilde{x}_3)$ ) with all but negligible probability.

*Proof.* Firstly, it follows from Lemma 4.2 that the view of  $Q$  in Scenario 1 comprising of  $\{\text{crs}, x_3, r_3, \mathbf{b}_{1 \rightarrow}^1, \mathbf{b}_{2 \rightarrow}^1, \mathbf{pc}_{1 \rightarrow 3}, \mathbf{b}_{1 \rightarrow}^2\}$  (see Table 3) results in output  $f(x_1, x_2, x_3)$  with all but negligible probability. Now, suppose Scenario 3 occurs, then  $P_1$  can locally emulate the view of  $Q$  at the end of Scenario 1 (w.r.t. different choices of inputs and randomness on behalf of  $P_1$  and  $P_3$  while the input of  $P_2$  remains fixed). Specifically, the view emulated by  $P_1$  for specific choice of  $(\widetilde{x}_1, \widetilde{x}_3)$  is identically distributed to the view of  $Q$  at the end of Scenario 1 if it occurred with respect to  $(\widetilde{x}_1, x_2, \widetilde{x}_3)$ . We can thus conclude that  $P_1$  can carry out output computation (similar to  $Q$  in Scenario 1) to learn  $f(\widetilde{x}_1, x_2, \widetilde{x}_3)$  with all but negligible probability. Hence,  $\Pi$  is such that it allows a semi-honest  $P_1$  to learn  $f(\widetilde{x}_1, x_2, \widetilde{x}_3)$  for different choices of  $(\widetilde{x}_1, \widetilde{x}_3)$  with all but negligible probability.  $\square$

We note that the above attack by an adversary corrupting  $P_1$  breaches privacy of honest  $P_2$ . This is because corrupt  $P_1$  can learn  $f(\widetilde{x}_1 = (\widetilde{m}_0, \widetilde{m}_1), x_2 = b, \widetilde{x}_3 = \perp) = \widetilde{m}_b$ , which reveals  $x_2 = b$  to  $P_1$ . This must not be allowed as per ideal realization of  $f$  (based on which  $P_1$  is not allowed to learn  $b$ ). We have thus arrived at a contradiction to our assumption that the two-round protocol  $\Pi$  is secure.

Lastly, we show how the above proof can be extended for  $n \geq 3$  and  $n/3 \leq t < n/2$  using party partitioning technique [Lyn96]. Assume towards a contradiction, that there exists a two-round  $n$ -party solitary MPC  $\Pi'$  computing  $f$  that achieves **god** against  $t$  corruptions, where  $2t < n \leq 3t$ . Then,  $\Pi'$  can be transformed into a two-round three-party solitary MPC protocol  $\Pi$  achieving **god** against a single corruption as follows: Partition the set of  $n$  parties into three disjoint groups  $S_1, S_2$  and  $S_3$  of size  $t, t$  and  $(n - 2t)$  respectively. Let  $P_i$  ( $i \in [3]$ ) in  $\Pi$  emulate the steps of parties in  $S_i$  during  $\Pi'$ . It is easy to see that security of  $\Pi'$  implies security of  $\Pi$  (as corruption of single party  $P_i$  ( $i \in [3]$ ) during  $\Pi$  is analogous to corruption of upto  $t$  parties in  $S_i$  during  $\Pi'$ ). However, our proof argument above showed the impossibility of such a two-round three-party solitary MPC protocol  $\Pi$  achieving **god** against a single corruption. We have thus arrived at a contradiction, completing the proof of Theorem 4.1.  $\square$

**Circumventions of the Lower Bound.** Before concluding the section, we present a couple of interesting circumventions of our lower bound:

- *Using PKI:* We point that our lower bound can be circumvented in the presence of private setup such as public-key infrastructure (PKI) due to the following reason. If a setup such as PKI is established,  $Q$  may hold some private information unknown to  $P_1$  at the end of Round 1, such as the decryption of  $P_2$ 's Round 1 broadcast using its exclusive secret key. This may aid in output computation by  $Q$ ; thereby the argument about the residual attack by  $P_1$

([Lemma 4.3](#)) does not hold. In fact, a two-round protocol achieving `god` can be designed in the presence of CRS and PKI setup as demonstrated by the work of [[GLS15](#)].

- *Single-input functions*: Since our final contradiction relies on residual attack by  $P_1$ , it would not hold in case  $f$  is a single-input function i.e. involves inputs provided only by single party. We note that employing a protocol for single-input functions seems meaningful when the input holding party is different from the output receiving party  $Q$ . In such scenarios, when a function  $f(x)$  is to be computed involving input  $x$  from a party  $P_i \neq Q$ , we point that any existing two-round two-party secure computation protocol (also known as non-interactive secure computation) between  $P_i$  and  $Q$  can be employed [[IKO<sup>+</sup>11](#), [AMPR14](#), [CJS14](#), [MR17](#), [BGI<sup>+</sup>17](#)]. This would trivially achieve `god` as the failure of the non-interactive secure computation protocol when  $Q$  is honest implies that  $P_i$  is corrupt; enabling  $Q$  to simply compute  $f$  on default input of  $P_i$ .

## 4.2 Necessity of Broadcast in Round 1

Now we show that any three-round  $n$ -party solitary MPC with `god` against  $t$  corruptions must use broadcast channel in Round 1, where  $n/3 \leq t < n/2$ .

**Theorem 4.4.** *Assume parties have access to CRS and pairwise-private channels. Let  $n$  and  $t$  be positive integers such that  $n \geq 3$  and  $n/3 \leq t < n/2$ . There exists a solitary functionality  $f$  such that no three-round  $n$ -party solitary MPC protocol securely computes  $f$  with `god` against  $t$  corruptions, while making use of the broadcast channel only in Round 2 and Round 3 (pairwise-private channels can be used in all the rounds).*

*Proof.* For simplicity, we present the argument for the setting  $n = 3$  and  $t = 1$  below. The proof can be extended for  $n/3 \leq t < n/2$  using player partitioning technique (as elaborated in the proof of [Theorem 4.1](#)). Suppose for the sake of contradiction that there exists a three-round solitary MPC protocol with `god`, say  $\Pi$  that utilizes broadcast channel only in Rounds 2 and 3 (i.e.,  $\Pi$  uses only pairwise-private channels in Round 1, and uses both broadcast and pairwise-private channels in Rounds 2 and 3).

Without loss of generality, we can assume that  $\Pi$  has the following structure: **(a)** No broadcast messages are sent during Round 3, and Round 3 only involves private messages sent to  $Q$ . This is without loss of generality as any solitary MPC that uses broadcast in the last round can be transformed into one where the messages sent via broadcast are sent privately only to  $Q$  (as  $Q$  is the only party supposed to receive output at the end of Round 3). **(b)** Round 2 only involves broadcast messages. This is also without loss of generality since the parties can perform pairwise-private communication by exchanging random pads in the first round and then using these random pads to unmask later broadcasts [[GIKR01](#)].

Let  $\Pi$  compute the solitary function  $f(x_1, x_2, x_3)$  among parties  $\{P_1, P_2, P_3\}$  where  $Q := P_3$  denotes the output receiving party. For simplicity, we use the same definition of  $f$  as in [Theorem 4.1](#) i.e.  $f(x_1 = (m_0, m_1), x_2 = b, x_3 = \perp) := m_b$ , where  $x_3 = \perp$  denotes that  $Q$  has no input;  $(m_0, m_1) \in \{0, 1\}^\lambda$  denote a pair of strings and  $b \in \{0, 1\}$  denotes a single bit. We analyze three different scenarios of the execution of  $\Pi$ . Before describing the scenarios, we define some useful notation. We assume  $(r_1, r_2, r_3)$  are the randomness used by the three parties if they behave honestly during the protocol execution. Let  $\text{pc}_{i \rightarrow j}$  where  $i, j \in [3]$  denote the pairwise-private communication from  $P_i$  to  $P_j$  in Round 1 if  $P_i$  behaves honestly using input  $x_i$  and randomness

$r_i$ . Similarly, let  $\widetilde{\text{pc}}_{i \rightarrow j}$  denote the pairwise-private communication from  $P_i$  to  $P_j$  in Round 1 if  $P_i$  follows the protocol but uses some other input  $\widetilde{x}_i$  and randomness  $\widetilde{r}_i$ . Let  $\mathbf{b}_i^{x,r,\text{pc}_{i-1},\text{pc}_{i+1}}$  where  $i \in [3]$  denote the broadcast communication by  $P_i$  in Round 2 if  $P_i$  behaves honestly using input  $x$  and randomness  $r$ , and received  $\text{pc}_{i-1}$  from  $P_{i-1}$  and  $\text{pc}_{i+1}$  from  $P_{i+1}$  in Round 1 (let  $P_0 := P_3$  and  $P_4 := P_1$ ). Lastly, let  $\text{pc}_{i \rightarrow 3}^\ell$  where  $i \in [2], \ell \in [3]$  denote the pairwise-private communication from  $P_i$  to  $Q$  in Round 3 in Scenario  $\ell$ . A party's view consists of  $\text{crs}$ , its input, randomness and incoming messages. Following is a description of the three scenarios. The views of the parties are described in Tables 4 – 5.

**Scenario 1:** Adversary corrupts  $P_2$ . In Round 1,  $P_2$  behaves honestly to  $P_1$  using input  $x_2$  and randomness  $r_2$  while behaving dishonestly to  $Q$  using  $(\widetilde{x}_2, \widetilde{r}_2)$ . In other words,  $P_2$  sends  $\text{pc}_{2 \rightarrow 1}$  to  $P_1$  and  $\widetilde{\text{pc}}_{2 \rightarrow 3}$  to  $Q$ .

In Round 2,  $P_2$  broadcasts a message as if he behaved honestly in Round 1 to both parties (using  $(x_2, r_2)$ ) and received a message from  $Q$  computed using  $(\widetilde{x}_3 = \perp, \widetilde{r}_3)$  in Round 1. Formally,  $P_2$  broadcasts  $\mathbf{b}_2^{x_2, r_2, \text{pc}_{1 \rightarrow 2}, \widetilde{\text{pc}}_{3 \rightarrow 2}}$ .

In Round 3,  $P_2$  aborts.

**Scenario 2:** Adversary corrupts  $Q$ . In Round 1,  $Q$  behaves towards  $P_1$  using  $(x_3 = \perp, r_3)$  while behaving towards  $P_2$  using  $(\widetilde{x}_3 = \perp, \widetilde{r}_3)$ . In other words,  $Q$  sends  $\text{pc}_{3 \rightarrow 1}$  to  $P_1$  and  $\widetilde{\text{pc}}_{3 \rightarrow 2}$  to  $P_2$ .

In Round 2,  $Q$  broadcasts a message as if he behaved honestly in Round 1 to both parties (using  $(x_3 = \perp, r_3)$ ) and received a message from  $P_2$  in Round 1 using  $(\widetilde{x}_2, \widetilde{r}_2)$ . Formally,  $Q$  broadcasts  $\mathbf{b}_3^{x_3, r_3, \text{pc}_{1 \rightarrow 3}, \widetilde{\text{pc}}_{2 \rightarrow 3}}$ .

**Scenario 3:** Adversary passively corrupts  $P_1$  behaving honestly using  $(x_1, r_1)$  in all rounds.

	Scenario 1			Scenario 2		
	View <sub>1</sub>	View <sub>2</sub>	View <sub>3</sub>	View <sub>1</sub>	View <sub>2</sub>	View <sub>3</sub>
Initial Input	$(x_1, r_1, \text{crs})$	$(x_2, r_2, \text{crs})$	$(x_3 = \perp, r_3, \text{crs})$	$(x_1, r_1, \text{crs})$	$(x_2, r_2, \text{crs})$	$(x_3 = \perp, r_3, \text{crs})$
Round 1	$\text{pc}_{2 \rightarrow 1}, \text{pc}_{3 \rightarrow 1}$	$\text{pc}_{1 \rightarrow 2}, \text{pc}_{3 \rightarrow 2}$	$\text{pc}_{1 \rightarrow 3}, \widetilde{\text{pc}}_{2 \rightarrow 3}$	$\text{pc}_{2 \rightarrow 1}, \text{pc}_{3 \rightarrow 1}$	$\text{pc}_{1 \rightarrow 2}, \widetilde{\text{pc}}_{3 \rightarrow 2}$	$\text{pc}_{1 \rightarrow 3}, \text{pc}_{2 \rightarrow 3}$
Round 2	$\mathbf{b}_2^{x_2, r_2, \text{pc}_{1 \rightarrow 2}, \widetilde{\text{pc}}_{3 \rightarrow 2}}$ $\mathbf{b}_3^{x_3, r_3, \text{pc}_{1 \rightarrow 3}, \widetilde{\text{pc}}_{2 \rightarrow 3}}$	$\mathbf{b}_1^{x_1, r_1, \text{pc}_{2 \rightarrow 1}, \text{pc}_{3 \rightarrow 1}}$ $\mathbf{b}_3^{x_3, r_3, \text{pc}_{1 \rightarrow 3}, \widetilde{\text{pc}}_{2 \rightarrow 3}}$	$\mathbf{b}_1^{x_1, r_1, \text{pc}_{2 \rightarrow 1}, \text{pc}_{3 \rightarrow 1}}$ $\mathbf{b}_2^{x_2, r_2, \text{pc}_{1 \rightarrow 2}, \widetilde{\text{pc}}_{3 \rightarrow 2}}$	$\mathbf{b}_2^{x_2, r_2, \text{pc}_{1 \rightarrow 2}, \widetilde{\text{pc}}_{3 \rightarrow 2}}$ $\mathbf{b}_3^{x_3, r_3, \text{pc}_{1 \rightarrow 3}, \widetilde{\text{pc}}_{2 \rightarrow 3}}$	$\mathbf{b}_1^{x_1, r_1, \text{pc}_{2 \rightarrow 1}, \text{pc}_{3 \rightarrow 1}}$ $\mathbf{b}_3^{x_3, r_3, \text{pc}_{1 \rightarrow 3}, \widetilde{\text{pc}}_{2 \rightarrow 3}}$	$\mathbf{b}_1^{x_1, r_1, \text{pc}_{2 \rightarrow 1}, \text{pc}_{3 \rightarrow 1}}$ $\mathbf{b}_2^{x_2, r_2, \text{pc}_{1 \rightarrow 2}, \widetilde{\text{pc}}_{3 \rightarrow 2}}$
Round 3	–	–	$\text{pc}_{1 \rightarrow 3}^1$	–	–	$\text{pc}_{1 \rightarrow 3}^2, \text{pc}_{2 \rightarrow 3}^2$

Table 4: Views of  $\{P_1, P_2, Q\}$  in Scenarios 1 and 2.

	View <sub>1</sub>	View <sub>2</sub>	View <sub>3</sub>
Initial Input	$(x_1, r_1, \text{crs})$	$(x_2, r_2, \text{crs})$	$(x_3 = \perp, r_3, \text{crs})$
Round 1	$\text{pc}_{2 \rightarrow 1}, \text{pc}_{3 \rightarrow 1}$	$\text{pc}_{1 \rightarrow 2}, \text{pc}_{3 \rightarrow 2}$	$\text{pc}_{1 \rightarrow 3}, \text{pc}_{2 \rightarrow 3}$
Round 2	$\mathbf{b}_2^{x_2, r_2, \text{pc}_{1 \rightarrow 2}, \text{pc}_{3 \rightarrow 2}}$ $\mathbf{b}_3^{x_3, r_3, \text{pc}_{1 \rightarrow 3}, \text{pc}_{2 \rightarrow 3}}$	$\mathbf{b}_1^{x_1, r_1, \text{pc}_{2 \rightarrow 1}, \text{pc}_{3 \rightarrow 1}}$ $\mathbf{b}_3^{x_3, r_3, \text{pc}_{1 \rightarrow 3}, \text{pc}_{2 \rightarrow 3}}$	$\mathbf{b}_1^{x_1, r_1, \text{pc}_{2 \rightarrow 1}, \text{pc}_{3 \rightarrow 1}}$ $\mathbf{b}_2^{x_2, r_2, \text{pc}_{1 \rightarrow 2}, \text{pc}_{3 \rightarrow 2}}$
Round 3	–	–	$\text{pc}_{1 \rightarrow 3}^3, \text{pc}_{2 \rightarrow 3}^3$

Table 5: Views of  $\{P_1, P_2, Q\}$  in Scenario 3.

The proof skeleton is as follows. First, we claim if Scenario 1 occurs, then  $Q$  must obtain  $f(x_1, x_2, \perp)$  with overwhelming probability. Due to the god property of  $\Pi$ , the honest  $Q$  in Scenario 1 must learn an output on the honest  $P_1$ 's input, namely  $x_1$ . The output should also be computed on  $P_2$ 's honest input  $x_2$  because  $Q$ 's view in Scenario 1 is subsumed by its view in Scenario 2, where the malicious  $Q$  can only learn an output computed on the honest  $P_2$ 's input. Intuitively,  $P_2$ 's input is “committed” in its private communication to  $P_1$  in Round 1 and broadcast message in Round 2. This allows a semi-honest  $P_1$  in Scenario 3 to emulate  $Q$ 's view in Scenario 1 and learn  $f(x_1, x_2, \perp)$ , which compromises the security of  $\Pi$ . We now prove a sequence of lemmas to complete our proof.

**Lemma 4.5.**  $\Pi$  must be such that if Scenario 1 or 2 occurs, then the output obtained by  $Q$  must be  $y = f(x_1, x_2, \perp)$  with all but negligible probability.

*Proof.* First, in Scenario 1, it follows from the god property of  $\Pi$  that  $Q$  obtains an output with all but negligible probability. Correctness dictates that the output must be computed w.r.t the honest input of  $P_1$ , namely  $x_1 = (m_0, m_1)$ . Therefore, the output should be  $f(x_1, x'_2, \perp) = m_{b'}$  for some  $x'_2 = b'$ . Observe that the message sent from  $P_1$  to  $Q$  in Round 3 is identically distributed in Scenarios 1 and 2, namely  $\widetilde{\text{pc}}_{1 \rightarrow 3}^2 = \text{pc}_{1 \rightarrow 3}^1$ . Hence the view of  $Q$  in Scenario 2 subsumes its view in Scenario 1 except  $\widetilde{\text{pc}}_{2 \rightarrow 3}$ . Notice that  $\widetilde{\text{pc}}_{2 \rightarrow 3}$  is computed using  $(\widetilde{x}_2, \widetilde{r}_2)$ , which can be arbitrarily chosen by  $Q$  and computed by  $Q$  himself. Therefore, a corrupted  $Q$  in Scenario 2 can emulate its view in Scenario 1, hence obtaining the output  $f(x_1, x'_2, \perp) = m_{b'}$ . By the security guarantee of  $\Pi$ , corrupted  $Q$  in Scenario 2 is not allowed to learn anything beyond the output computed w.r.t the honest inputs of  $P_1$  and  $P_2$  i.e. the output  $f(x_1, x_2, \perp) = m_b$  for  $x_2 = b$ . We can thus conclude that  $b' = b$  must hold with all but negligible probability and thus, in both Scenarios 1 and 2, the output obtained by  $Q$  must be  $y = f(x_1, x_2, \perp)$  with all but negligible probability.  $\square$

**Lemma 4.6.**  $\Pi$  must be such that if Scenario 3 occurs, then the adversary  $P_1$  can learn  $f(x_1, x_2, \perp)$  with all but negligible probability.

*Proof.* Consider Scenario 1 with  $(x_3 = \perp, r_3)$  and  $(\widetilde{x}_3 = \perp, \widetilde{r}_3)$  flipped. We call this Scenario 4 and present the views of the three parties in Table 6.

	View <sub>1</sub>	View <sub>2</sub>	View <sub>3</sub>
Initial Input	$(x_1, r_1, \text{crs})$	$(x_2, r_2, \text{crs})$	$(\widetilde{x}_3 = \perp, \widetilde{r}_3, \text{crs})$
Round 1	$\widetilde{\text{pc}}_{2 \rightarrow 1}, \widetilde{\text{pc}}_{3 \rightarrow 1}$	$\text{pc}_{1 \rightarrow 2}, \text{pc}_{3 \rightarrow 2}$	$\text{pc}_{1 \rightarrow 3}, \widetilde{\text{pc}}_{2 \rightarrow 3}$
Round 2	$b_2^{x_2, r_2, \text{pc}_{1 \rightarrow 2}, \text{pc}_{3 \rightarrow 2}}$ $b_3^{\widetilde{x}_3, \widetilde{r}_3, \text{pc}_{1 \rightarrow 3}, \widetilde{\text{pc}}_{2 \rightarrow 3}}$	$b_1^{x_1, r_1, \text{pc}_{2 \rightarrow 1}, \widetilde{\text{pc}}_{3 \rightarrow 1}}$ $b_3^{\widetilde{x}_3, \widetilde{r}_3, \text{pc}_{1 \rightarrow 3}, \widetilde{\text{pc}}_{2 \rightarrow 3}}$	$b_1^{x_1, r_1, \text{pc}_{2 \rightarrow 1}, \widetilde{\text{pc}}_{3 \rightarrow 1}}$ $b_2^{x_2, r_2, \text{pc}_{1 \rightarrow 2}, \text{pc}_{3 \rightarrow 2}}$
Round 3	–	–	$\text{pc}_{1 \rightarrow 3}^4$

Table 6: Views of  $\{P_1, P_2, Q\}$  in Scenario 4.

By Lemma 4.5,  $Q$  should learn  $f(x_1, x_2, \perp)$  in Scenario 4. We now show that  $P_1$  and  $Q$ 's views in Scenario 4 can be emulated by corrupted  $P_1$  in Scenario 3.

First,  $P_1$  knows  $(x_1, r_1, \text{crs})$  and he can choose arbitrary  $(\widetilde{x}_2, \widetilde{r}_2)$  for  $P_2$ , set  $\widetilde{x}_3 = \perp$  and choose arbitrary  $\widetilde{r}_3$  for  $Q$  as the initial inputs. In Round 1,  $\text{pc}_{2 \rightarrow 1}$  is the message  $P_1$  receives from  $P_2$  in Scenario 3.  $\widetilde{\text{pc}}_{3 \rightarrow 1}$  can be computed using  $(\widetilde{x}_3 = \perp, \widetilde{r}_3, \text{crs})$ .  $\text{pc}_{1 \rightarrow 3}$  can be computed using

$(x_1, r_1, \text{crs})$ .  $\widetilde{\text{pc}}_{2 \rightarrow 3}$  can be computed using  $(\widetilde{x}_2, \widetilde{r}_2, \text{crs})$ . In Round 2,  $\text{b}_2^{x_2, r_2, \text{pc}_{1 \rightarrow 2}, \text{pc}_{3 \rightarrow 2}}$  is the message broadcast by  $P_2$  in Round 2 of Scenario 3.  $\text{b}_3^{\widetilde{x}_3, \widetilde{r}_3, \text{pc}_{1 \rightarrow 3}, \text{pc}_{2 \rightarrow 3}}$  can be computed from  $(\widetilde{x}_3, \widetilde{r}_3, \text{crs})$ ,  $\text{pc}_{1 \rightarrow 3}$  (which can be computed using  $(x_1, r_1, \text{crs})$ ),  $\widetilde{\text{pc}}_{2 \rightarrow 3}$  (which can be computed using  $(\widetilde{x}_2, \widetilde{r}_2, \text{crs})$ ).  $\text{b}_1^{x_1, r_1, \text{pc}_{2 \rightarrow 1}, \text{pc}_{3 \rightarrow 1}}$  can be computed from  $(x_1, r_1, \text{crs})$ ,  $\text{pc}_{2 \rightarrow 1}$ ,  $\widetilde{\text{pc}}_{3 \rightarrow 1}$ . In Round 3,  $\text{pc}_{1 \rightarrow 3}^4$  can be computed from  $P_1$ 's view in Scenario 4.

Therefore,  $P_1$  can emulate  $Q$ 's view in Scenario 4 and learn  $f(x_1, x_2, \perp)$ .  $\square$

By Lemma 4.6, the corrupted  $P_1$  can learn  $f(x_1 = (m_0, m_1), x_2 = b, \perp) = m_b$  which allows  $P_1$  to learn honest  $P_2$ 's input  $x_2 = b$ . This contradicts with the security guarantee of  $\Pi$  based on which parties other than  $Q$  must learn no information. Note that since  $f$  does not involve an input from  $Q$ , our proof shows that this argument holds irrespective of whether  $Q$  has an input or not.  $\square$

### 4.3 Necessity of Broadcast in Round 2

In this section, we show that any three-round  $n$ -party solitary MPC with **god** against  $t$  corruptions must use broadcast channel in Round 2 when  $2 \lceil n/5 \rceil \leq t < n/2$  (note that  $t \geq 2$ ). Interestingly, the use of broadcast in Round 2 is not necessary for the special case of single corruption, which we discuss in Section 4.4.

**Theorem 4.7.** *Assume parties have access to CRS. Let  $n$  and  $t$  be positive integers such that  $n \geq 5$  and  $2 \lceil n/5 \rceil \leq t < n/2$ . Then, there exists a solitary functionality  $f$  such that no three-round  $n$ -party solitary MPC protocol tolerating  $t$  corruptions securely computes  $f$  with **god**, while making use of the broadcast channel only in Round 1 and Round 3 (pairwise-private channels can be used in all the rounds).*

*Proof.* We present the argument for the setting of  $n = 5$  and  $t = 2$  below, and elaborate later on how to extend to  $2 \lceil n/5 \rceil \leq t < n/2$ . Suppose for the sake of contradiction that there exists a three-round 5-party solitary MPC protocol with **god** against two corruptions, say  $\Pi$  that utilizes broadcast channel only in Round 1 and Round 3 (i.e.  $\Pi$  uses broadcast and pairwise-private channels in Round 1 and Round 3; and only pairwise-private channels in Round 2).

Without loss of generality, we assume for simplicity that  $\Pi$  has the following structure: **(a)** No broadcast messages are sent during Round 3 and Round 3 involves only private messages sent to  $Q$ . This is w.l.o.g as any solitary MPC that uses broadcast in last round can be transformed to one where the messages sent via broadcast are sent privately only to  $Q$  (as  $Q$  is the only party supposed to receive output at the end of Round 3). **(b)** Round 2 does not involve messages from  $P_i$  ( $i \in [4]$ ) to  $Q$  (such a message is meaningful only if  $Q$  communicates to  $P_i$  in Round 3, which is not the case as per **(a)**).

Let  $\Pi$  compute the solitary function  $f(x_1, \dots, x_5)$  among  $\{P_1, \dots, P_5\}$  where  $Q = P_5$  denotes the output receiving party. We clarify that our argument holds irrespective of whether  $f$  involves an input from  $Q$  or not. Let  $f(x_1 = (x_r, x_c), x_2 = (x_2^0, x_2^1), x_3 = (x_3^0, x_3^1), x_4 = \perp, x_5 = \perp)$  with  $x_1, x_2, x_3 \in \{0, 1\}^2$  be defined as

$$f(x_1, \dots, x_5) = \begin{cases} (x_r \oplus x_2^0, x_3^0) & \text{if } x_c = 0 \\ (x_r \oplus x_2^1, x_3^1) & \text{if } x_c = 1 \end{cases} .$$

We consider an execution of  $\Pi$  with inputs  $(x_1, \dots, x_5)$  where  $x_i$  denotes the input of  $P_i$ . In the above definition of  $f$ ,  $x_4 = x_5 = \perp$  indicates that  $P_4$  and  $P_5$  do not have any inputs. Next, we

analyze four different scenarios. Before describing the scenarios, we define some useful notation. Let  $\mathbf{b}_i^1$  denote the broadcast communication by  $P_i$  in Round 1 when  $P_i$  behaves honestly. In Rounds 1 and 2, let  $\mathbf{pc}_{i \rightarrow j}^r$ , where  $r \in [2]$ ,  $i, j \in [5]$  denote the pairwise-private communication from  $P_i$  to  $P_j$  in Round  $r$ , as per an execution where everyone behaves honestly. Next, we use  $\widetilde{\mathbf{pc}_{i \rightarrow j}^2}$  to denote the messages that  $P_i$  ( $i \in [5]$ ) is supposed to send in Round 2 to  $P_j$  ( $j \in [4] \setminus i$ ) incase  $P_i$  did not receive Round 1 message from  $P_1$ . Note that this communication could be potentially different from what  $P_i$  would send in an honest execution. Lastly, since Round 3 messages to  $Q$  could potentially be different for each of the four scenarios, we index them additionally with  $\ell$  indicating the scenario i.e.  $\mathbf{pc}_{j \rightarrow 5}^{3, \ell}$  denotes  $P_j$ 's Round 3 message to  $Q$  in Scenario  $\ell$  ( $j \in [4]$ ,  $\ell \in [4]$ ). These messages may be a function of the common reference string (denoted by  $\text{crs}$ ). A party's view comprises of  $\text{crs}$ , its input, randomness and incoming messages.

Following is a description of the scenarios. In each of these scenarios, we assume that the adversary uses the honest input on behalf of the corrupt parties and its malicious behaviour is limited to dropping some of the messages that were received or supposed to be sent by the actively corrupt parties. The views of the parties are described in Tables 7 – 10.

- Scenario 1:** Adversary corrupts  $P_1$ . In Round 1,  $P_1$  behaves honestly w.r.t his broadcast communication and private message towards  $P_2$  and  $Q$ , but drops his private message towards  $P_3$  and  $P_4$ . Further,  $P_1$  remains silent after Round 1 (i.e. does not communicate at all in Round 2 and Round 3). In other words, in Scenario 1,  $P_1$  computes and sends only the following messages honestly :  $\mathbf{b}_1^1$ ,  $\mathbf{pc}_{1 \rightarrow 2}^1$  and  $\mathbf{pc}_{1 \rightarrow 5}^1$ .
- Scenario 2:** Adversary corrupts  $\{P_1, P_2\}$ .  $P_1$  behaves identical to Scenario 1.  $P_2$  behaves honestly except that he drops his Round 3 message towards  $Q$ .
- Scenario 3:** Adversary corrupts  $\{P_3, P_4\}$ . In Round 1,  $\{P_3, P_4\}$  behave honestly as per protocol steps. In Round 2,  $\{P_3, P_4\}$  only communicate to  $P_2$ , towards whom they pretend that they did not receive Round 1 message from  $P_1$  (i.e.  $P_i$  sends  $\widetilde{\mathbf{pc}_{i \rightarrow 2}^2}$  to  $P_2$  where  $i \in \{3, 4\}$ ). Lastly,  $\{P_3, P_4\}$  remain silent in Round 3 i.e. do not communicate towards  $Q$ .
- Scenario 4:** Adversary corrupts  $\{P_1, Q\}$ .  $Q$  behaves honestly throughout the protocol.  $P_1$  behaves as follows: In Round 1,  $P_1$  behaves identical to Scenario 1 (i.e. behaves honestly w.r.t its broadcast communication and private message to  $P_2$  and  $Q$ ; but drops his private message to  $P_3$  and  $P_4$ ). In Round 2,  $P_1$  behaves honestly only to  $P_2$  (but does not communicate to others). Lastly,  $P_1$  sends its Round 3 message to  $Q$  as per Scenario 3 (i.e. as per protocol specifications when  $P_1$  does not receive Round 2 message from  $P_3$  and  $P_4$ ). The communication in Round 3 among the corrupt parties is mentioned only for clarity.



The proof skeleton is as follows. First, we claim that there exists an  $x_c^* \in \{0, 1\}$  such that if Scenario 1 occurs with respect to  $x_c = x_c^*$ , then the output obtained by  $Q$  must be computed with respect to  $\neg x_c^*$  with non-negligible probability. Intuitively, if for all  $x_c$ , the output of Scenario 1 was computed on  $x_c$ , then it would mean that  $\{P_2, Q\}$  have sufficient information about  $x_c$  at the end of Round 1 itself. This would make  $\Pi$  vulnerable to a residual function attack by  $\{P_2, Q\}$ . Next, we claim that if  $x_c = x_c^*$ , even the output of Scenario 2 must be computed on  $\neg x_c^*$  with non-negligible probability. Regarding Scenario 3, correctness of  $\Pi$  lets us infer that  $Q$  must compute output on the input  $x_1 = (x_r, x_c)$  of honest  $P_1$ . Lastly, we argue that  $Q$ 's view in Scenario 4 subsumes its views in Scenario 2 and Scenario 3. This would allow corrupt  $\{P_1, Q\}$  (who participate with  $x_c = x_c^*$ ) in Scenario 4 to obtain multiple outputs i.e. output with respect to both  $\neg x_c^*$  (as in Scenario 2) and  $x_c^*$  (as in Scenario 3), which contradicts security of  $\Pi$ . This completes the proof sketch. We present below a sequence of inferences to formalize the above proof sketch.

**Lemma 4.8.** *There exists an  $x_c^* \in \{0, 1\}$  such that if Scenario 1 of  $\Pi$  occurs with respect to  $x_c = x_c^*$ , then the output obtained by  $Q$  must be  $f(x'_1 = (x'_r, x'_c), x_2, x_3, x_4, x_5)$  where  $x'_c \neq x_c^*$  (namely,  $x'_c = \neg x_c^*$ ) with non-negligible probability.*

*Proof.* Suppose Scenario 1 occurs. First, it follows from the god property of  $\Pi$  that  $Q$  obtains an output with overwhelming probability. Correctness dictates that the output should be computed w.r.t honest inputs of  $\{P_2, P_3, P_4, P_5\}$ . Therefore,  $Q$  obtains  $f(x'_1, x_2, x_3, x_4, x_5)$  for some  $x'_1 = (x'_r, x'_c)$ .

Assume towards a contradiction that for all  $x_c \in \{0, 1\}$ ,  $x'_c \neq x_c$  with negligible probability; namely  $x'_c = x_c$  with overwhelming probability. Then, we demonstrate below an adversarial strategy that breaches security of  $\Pi$ .

Consider a scenario (say  $S^*$ ) where an adversary corrupts  $\{P_2, Q\}$  passively and honest  $P_1$  participates with input  $x_1 = (x_r, x_c)$ . We claim that the adversary can compute the output on  $x_c$  i.e.  $f(x'_1 = (x'_r, x_c), x'_2, x'_3, x'_4 = \perp, x'_5 = \perp)$  for any choice of  $(x'_2, x'_3)$  with overwhelming probability. This can be done by emulating Scenario 1 as follows - The adversary chooses the set of inputs  $x'_2$  and  $x'_3$  on behalf of  $\{P_2, P_3\}$  and randomness on behalf of  $\{P_2, P_3, P_4, P_5\}$ . Next, he fixes the messages  $\mathbf{b}_1^1$ ,  $\text{pc}_{1 \rightarrow 2}^1$  and  $\text{pc}_{1 \rightarrow 5}^1$  as received from  $P_1$  in Round 1 (on behalf of  $\{P_2, Q\}$ ). Recall that  $\{\mathbf{b}_1^1, \text{pc}_{1 \rightarrow 2}^1, \text{pc}_{1 \rightarrow 5}^1\}$  constitutes the only communication from  $P_1$  in Scenario 1. Therefore, the adversary in  $S^*$  can locally compute a view that is identically distributed to the view of an honest  $Q$  in Scenario 1 w.r.t set of inputs  $(x_1 = (x_r, x_c), x'_2, x'_3, x'_4 = \perp, x'_5 = \perp)$ .

Based on our assumption, this view would allow the adversary to locally compute the output with respect to  $x_c$  i.e.  $f(x'_1 = (x'_r, x_c), x'_2 = (x_2^{0'}, x_2^{1'}), x'_3 = (x_3^{0'}, x_3^{1'}), \perp, \perp)$  with overwhelming probability. This breaches privacy of honest  $P_1$  in  $S^*$ , as the adversary can learn  $x_c$  by setting the chosen inputs appropriately (say, by choosing  $x_3^{0'} \neq x_3^{1'}$ ). Note that as per the ideal realization of  $f$ ,  $\{P_2, Q\}$  cannot infer  $x_c$  from the output. Thus, we have arrived at a contradiction; completing the proof.  $\square$

The above lemma shows that there exists an  $x_c^*$  such that if Scenario 1 occurs with respect to  $x_c = x_c^*$ , then the output would be computed on  $x'_c = \neg x_c^*$  with non-negligible probability. We use this particular  $x_c^*$  in the rest of the proof, where  $x_c^*$  could be either 0 or 1 and can be guessed correctly by the adversary with probability at least  $1/2$ .

**Lemma 4.9.**  *$\Pi$  must be such that if Scenario 2 occurs and  $x_c = x_c^*$ , then the output obtained by  $Q$  must be computed with respect to  $x'_c = \neg x_c^*$  and  $x_3$  with non-negligible probability.*

*Proof.* It is easy to see that **god** and correctness properties imply that  $Q$  must receive an output on honest  $P_3$ 's input i.e.  $x_3$  with overwhelming probability. Next, we focus on  $x'_c$  with respect to which the output is computed.

We observe that Scenario 1 and 2 proceed identically until Round 3. The only difference is that  $P_2$  drops its Round 3 message to  $Q$ . Thereby, we can infer that the view of  $Q$  in Scenario 2 is subsumed by its view in Scenario 1 (refer Tables 7 - 8).

Towards a contradiction, assume  $Q$  obtains an output computed on  $x'_c = \neg x_c^*$  with negligible probability, namely  $x'_c = x_c^*$  with overwhelming probability. Then there exists an adversarial strategy that allows the adversary to get multiple evaluations of the function  $f$  - Consider a scenario  $S'$  where the adversary corrupts  $\{P_1, Q\}$ . Suppose misbehaviour of corrupt  $P_1$  is identical to Scenario 1 and  $Q$  is passively corrupt. Since view of  $Q$  in Scenario  $S'$  is identically distributed to its view in Scenario 1, he can obtain an output computed on  $\neg x_c^*$  and  $(x_2, x_3)$  with non-negligible probability (Lemma 4.8). Further, passive  $Q$  can emulate Scenario 2 by simply discarding the Round 3 message from  $P_2$  in Scenario  $S'$ . This will allow the adversary to obtain the output of Scenario 2 that is computed on  $x'_c = x_c^*$  with overwhelming probability (as per our assumption) and  $x_3$  as well. This contradicts the security of  $\Pi$ . In particular, corrupted  $\{P_1, Q\}$  in Scenario  $S'$  can learn both  $x_3^0$  and  $x_3^1$  with non-negligible probability, which is not allowed as per the definition of the function  $f$ . We can thus conclude that the output of Scenario 2 is computed on  $x'_c = \neg x_c^*$  with non-negligible probability.  $\square$

**Lemma 4.10.**  *$\Pi$  must be such that if Scenario 3 occurs, then the output obtained by  $Q$  must be computed on  $x_1$  and  $x_2$  with overwhelming probability.*

*Proof.* Suppose Scenario 3 occurs. Since  $P_1, P_2$  and  $Q$  are honest, it follows from properties of **god** and correctness of  $\Pi$  that  $Q$  obtains an output computed on honest  $P_1$  and  $P_2$ 's inputs i.e.  $x_1$  and  $x_2$  with overwhelming probability.  $\square$

**Lemma 4.11.**  *$\Pi$  must be such that if Scenario 4 occurs with respect to  $x_c = x_c^*$ , then the adversary corrupting  $\{P_1, Q\}$  obtains multiple evaluations of  $f$  with non-negligible probability.*

*Proof.* Suppose Scenario 4 occurs. First, we claim that  $Q$  can obtain the output of Scenario 2 that is computed on  $\neg x_c^*$  and  $x_3$  (Lemmas 4.8 - 4.9) with non-negligible probability. Note that the Round 3 messages received by  $Q$  from  $\{P_3, P_4\}$  in Scenario 4 is identically distributed to the respective messages in Scenario 2 (as views of  $\{P_3, P_4\}$  in Scenarios 2 and 4 are identically distributed). It is now easy to check that  $Q$ 's view in Scenario 4 subsumes its view in Scenario 2 (see Tables 8 and 10). Thus,  $Q$  must be able to learn an output computed on  $\neg x_c^*$  and  $x_3$  with non-negligible probability.

Next, we argue similarly that  $Q$  can obtain the output of Scenario 3 as well. This is because the Round 3 messages received by  $Q$  in Scenario 4 from corrupt  $P_1$  (who pretends as if he did not receive Round 2 message from  $\{P_3, P_4\}$ ) and honest  $P_2$  (whose view is identically distributed to its view in Scenario 3) is identically distributed to the respective messages in Scenario 3. It is now easy to check that  $Q$ 's view in Scenario 4 subsumes its view in Scenario 3 (see Tables 9 - 10). Thus,  $Q$  must be able to learn the output of Scenario 3 which is computed w.r.t  $x_1$  and  $x_2$  (Lemma 4.10) with overwhelming probability.

We can thus conclude that in Scenario 4,  $Q$  can obtain an output computed on  $x_1 = (x_r, x_c^*)$  and  $x_2$  as well as another output computed on  $\neg x_c^*$  and  $x_3$ , with non-negligible probability.  $\square$

According to [Lemma 4.11](#), there exists an adversarial strategy that allows an adversary corrupting  $\{P_1, Q\}$  to obtain multiple evaluations of  $f$ , while the inputs of honest parties  $P_2, P_3$  and  $P_4$  remains fixed. Specifically, if  $x_c^* = 0$ , then the adversary learns both  $x_r \oplus x_2^0$  and  $x_3^1$  with non-negligible probability, from which  $x_2^0$  can be inferred (as  $x_r$  is known to adversary in Scenario 4 corrupting  $P_1$ ). This is not allowed as per the definition of  $f$ . Similarly, if  $x_c^* = 1$ , then the adversary can learn both  $x_2^1$  and  $x_3^0$  with non-negligible probability, which is also not allowed. This gives us the final contradiction, completing the proof of [Theorem 4.7](#) for the setting  $n = 5$  and  $t = 2$ . Note that since  $f$  did not involve an input from  $Q$  (i.e.  $x_5 = \perp$ ), our argument holds irrespective of whether  $Q$  has input or not.

Lastly, we show how the above proof can be extended for  $2 \lceil n/5 \rceil \leq t < n/2$  using party partitioning technique [[Lyn96](#)]. Assume towards a contradiction, that there exists a three-round  $n$ -party solitary MPC  $\Pi'$  that achieves `god` against  $t$  corruptions without using broadcast in Round 2, where  $2t < n \leq 5 \lfloor t/2 \rfloor$  (equivalent to  $2 \lceil n/5 \rceil \leq t < n/2$ <sup>6</sup>) and  $t \geq 2$ . Then,  $\Pi'$  can be transformed to a three-round 5-party solitary MPC protocol  $\Pi$  that does not use broadcast in Round 2 and achieves `god` against 2 corruptions as follows: Partition the set of  $n$  parties into 5 disjoint groups, say  $S_i$  ( $i \in [5]$ ), where  $S_1, \dots, S_4$  are each of size  $\lfloor t/2 \rfloor$  and  $S_5$  comprises of the remaining  $(n - 4 \lfloor t/2 \rfloor)$  parties. Let  $P_i$  ( $i \in [5]$ ) in  $\Pi$  emulate the steps of parties in  $S_i$  during  $\Pi'$ . It is easy to see that security of  $\Pi'$  implies security of  $\Pi$  (as corruption of up to 2 parties in  $\Pi$  is analogous to corruption of up to  $t$  parties in  $\Pi'$ ). However, our proof argument above showed the impossibility of such a three-round 5-party solitary MPC protocol  $\Pi$  that does not use broadcast in Round 2 and achieves `god` against two corruptions. We have thus arrived at a contradiction, completing the proof of [Theorem 4.7](#). Note that for certain cases, such as  $n = 6$ , this range of values of  $(n, t)$  is not meaningful. However, this is relevant for sufficiently large values of  $n$ .  $\square$

## 4.4 Upper Bounds

In this section, we discuss how the existing results of [[GLS15, ACGJ18, BJMS20, PR18](#)] imply that our negative results in the previous sections are tight.

**Three-round solitary MPC.** With respect to our three-round lower bound in [Section 4.1](#), we note that the existing three-round upper bounds of [[GLS15, ACGJ18, BJMS20](#)] for standard MPC are optimal for solitary MPC as well. This holds since any standard MPC with global output (that gives same output to all) can be transformed to one with private output (which gives private output to selected parties, only  $Q$  in our context) [[MZ13](#)]. For the sake of completeness, we recall this transformation: Let  $C$  denote the private-output circuit (which gives output only to  $Q$ ) computing the solitary function  $f(x_1, \dots, x_n)$ . Then instead of evaluating  $C$ , the standard MPC protocols of [[GLS15, ACGJ18, BJMS20](#)] can be used to evaluate a circuit  $C'$  (with global output) which takes as input  $x_i$  from each  $P_i$  and additionally a random pad  $r$  from  $Q$  to be used for “masking” the output of  $Q$ .  $C'$  evaluates  $C$  using the received inputs to compute  $f(x_1, \dots, x_n) = y$  and outputs  $y \oplus r$  as the global output to all. This public output can be unmasked by  $Q$  to recover the desired private output  $y = f(x_1, \dots, x_n)$ .

**Broadcast-optimal three-round MPC** Next, recall that our negative results in [Section 4.2](#) and [Section 4.3](#) prove that when  $t \geq 2$ , use of broadcast in both Round 1 and Round 2 are

<sup>6</sup>Note that  $n \leq 5 \lfloor t/2 \rfloor \iff n/5 \leq \lfloor t/2 \rfloor \iff \lceil n/5 \rceil \leq t/2 \iff 2 \lceil n/5 \rceil \leq t$ .

necessary for a three-round solitary MPC with `god` in honest majority (in particular, when  $n \geq 5$  and  $2 \lceil n/5 \rceil \leq t < n/2$ ). This result is tight, as the last round messages of the three-round protocols of [GLS15, ACGJ18, BJMS20] can be sent over pairwise-private channels to the output-receiving party, to construct a solitary MPC with `god` in honest majority that uses broadcast only in Round 1 and Round 2.

Since standard MPC implies solitary MPC, we can infer that use of broadcast in both Round 1 and Round 2 are necessary for a three-round standard MPC with `god` as well, in the honest majority setting. In order to demonstrate tightness for standard MPC, we observe that the three-round standard MPC protocols of [GLS15, BJMS20] which use broadcast in all three rounds, would achieve `god` even if the last round broadcast message of every party was instead sent via pairwise-private channels to every other party. First, we recall the structure of the protocol in [BJMS20] but note that the work of [GLS15] has a very similar structure. The first round involves each party generating and broadcasting some public keys for setting up the threshold multi-key fully homomorphic encryption (TMFHE) scheme and shares of a common reference string for a multi-string NIZK (we refer the reader to [BJMS20] for a formal definition of both these primitives). In the second round, each party encrypts their input using the combined FHE public key and generates a NIZK that the ciphertext was honestly generated. The ciphertext and the NIZK is broadcasted by every party. At the end of Round 2, every party can combine the ciphertexts locally to compute an encryption of the evaluation of the function  $f$  on all parties' inputs. In Round 3, parties broadcast their partial decryptions of this evaluated ciphertext along with a NIZK proving that the partial decryption was honestly generated. Intuitively, `god` holds on setting the threshold in the TMFHE scheme to be  $(\frac{n}{2} + 1)$  as the scheme then requires only a majority of partial decryptions to correctly decrypt and learn the output, which is immediately ensured in the honest majority setting. We now observe that, in Round 3, even if the partial decryptions (and associated NIZKs) were sent over pairwise-private channels, since there is an honest majority, every honest party would receive at least  $(\frac{n}{2} + 1)$  validly generated partial decryptions and can recover the output. Any invalid partial decryption received from a corrupt party can be detected (via the NIZK) and discarded, without affecting the output reconstruction procedure.

**Special case:**  $t = 1$ . Lastly, we discuss the setting of  $t = 1$ . We observe that analyzing use of broadcast for protocols achieving `god` in this setting is relevant only when  $n = 3$ , as two-round protocols achieving `god` in the presence of just pairwise-private channels are known when  $n \geq 4$  and  $t = 1$  [IKP10, IKKP15]. For the setting of  $n = 3$  and  $t = 1$ , three rounds are known to be necessary and sufficient for `god` against single corruption [PR18] (even in the presence of CRS and broadcast channel) and use of broadcast is necessary in Round 1 of such a three-round protocol (Section 4.2). The necessity of broadcast in Round 1 is in fact tight even for standard MPC (which implies solitary MPC) when  $n = 3$ ; as there exists a three-Round 3-party standard MPC protocol [PR18] that achieves `god` against single corruption and uses broadcast only in Round 1. For the sake of completeness, we give a high-level sketch of their protocol, say  $\Pi$ .  $\Pi$  involves three parallel executions of a two-round sub-protocol, say  $\Pi_i$  ( $i \in [3]$ ), where  $P_i$  is referred to as the evaluator and the other two parties are referred to as the garblers. During Round 1 of  $\Pi_i$ , each garbler  $P_j$  ( $j \neq i$ ) samples randomness and broadcasts a randomized encoding of the function to be computed (i.e. the garbled circuit), while sharing the randomness privately with its co-garbler. Additionally,  $P_i$  splits its input into two shares and sends one share to each garbler privately. Each garbler verifies the garbled circuit sent by its co-garbler over broadcast channel (using the

randomness received privately). In Round 2, a garbler  $P_j$  sends the encoded input to  $P_i$  (over pairwise-private channel) corresponding to its own input and the input share of  $P_i$  held by  $P_j$ . The garbler  $P_j$  would send the encoded input for its own garbled circuit (i.e. the one for which  $P_j$  sampled randomness) as well as for the garbled circuit of its co-garbler (if the verification passes). The invariant maintained is that  $P_i$  either obtains the function output or identifies a corrupt party, at the end of  $\Pi_i$ . In the former case,  $P_i$  sends the output over private channels to the other two parties. In the latter case, an honest  $P_i$  sends its input and the shares received in Round 1 of  $\Pi_j$  ( $j \neq i$ ) on clear to the other honest party (can be determined by  $P_i$  as the single corrupt party has been identified). This would enable correct output computation by honest parties. Apart from the above communication, some additional messages are sent privately in Round 2 to enforce input consistency in the sub-protocols. This completes the sketch of the construction of [PR18] that uses broadcast only in Round 1; thereby demonstrating tightness of our negative result in Section 4.2.

## 5 With PKI and No Broadcast

In this section, we consider the setting where the parties only have access to pairwise-private channels. In terms of setup, we assume that all parties have access to a public-key infrastructure (PKI) and a common reference string (CRS). We first present a lower bound of four rounds for solitary MPC with `god`. Then we present a five-round construction that works for any  $n$  and  $t < n/2$ . Next, we elaborate on a non-constant round protocol (i.e  $(t + 2)$  rounds) that can be derived from the protocol of [GLS15]. While the former upper bound significantly improves over the latter for most values of  $(n, t)$ , the latter achieves better round complexity for special cases of  $t \leq 2$ .

### 5.1 Necessity of Four Rounds

In this section, we assume a network setting where the parties have access to pairwise-private channels and PKI. We show that when  $3 \lceil n/7 \rceil \leq t < n/2$ , four rounds are necessary for  $n$ -party solitary MPC with `god` against  $t$  corruptions. This holds irrespective of whether  $Q$  has input or not and even if the adversary is non-rushing. However, the argument crucially relies on the fact that  $t \geq 3$  (details appear at the end of this section) which leads us to conjecture that there is a potential separation between the cases of  $t \leq 2$  and  $t \geq 3$  for solitary MPC. We investigate the special cases of  $t \leq 2$  in Section 6 and Section 7. The impossibility for the general case is formally stated below.

**Theorem 5.1.** *Assume parties have access to CRS, PKI and pairwise-private channels. Let  $n, t$  be positive integers such that  $n \geq 7$  and  $3 \lceil n/7 \rceil \leq t < n/2$ . Then, there exists an solitary functionality  $f$  such that no three-round  $n$ -party MPC protocol tolerating  $t$  corruptions can compute  $f$  with `god`, even if the adversary is assumed to be non-rushing.*

*Proof.* For simplicity, we consider the setting of  $n = 7$  and  $t = 3$  (extension to any  $3 \lceil n/7 \rceil \leq t < n/2$  appears later). Suppose for the sake of contradiction that there exists a three-round solitary MPC protocol with `god`, say  $\Pi$ . Let  $\Pi$  compute the solitary function  $f(x_1, \dots, x_7)$  among  $\{P_1, \dots, P_7\}$  where  $Q = P_7$  denotes the output receiving party. We clarify that our lower bound argument holds irrespective of whether  $f$  involves an input from  $Q$ . For simplicity, define  $f(x_1, x_2 = \perp, x_3 =$

$(x_3^0, x_3^1), x_4 = (x_4^0, x_4^1), x_5 = \perp, x_6 = (x_6^0, x_6^1), x_7 = \perp$ ) as

$$f(x_1, \dots, x_7) = \begin{cases} (x_3^0, x_4^0, x_6^0) & \text{if } x_1 = 0 \\ (x_3^1, x_4^1, x_6^1) & \text{if } x_1 = 1 \end{cases}$$

where  $x_1 \in \{0, 1\}$  and  $x_3, x_4, x_6 \in \{0, 1\}^2$ . In the definition,  $x_2 = x_5 = x_7 = \perp$  indicates that  $P_2, P_5, P_7$  do not have any inputs.

Without loss of generality, we assume for simplicity that  $\Pi$  has the following structure: **(a)** Round 3 involves only messages sent to  $Q$  (as  $Q$  is the only party supposed to receive output at the end of Round 3). **(b)** Round 2 does not involve messages from  $P_i$  ( $i \in [6]$ ) to  $Q$  (such a message is meaningful only if  $Q$  communicates to  $P_i$  in Round 3, which is not the case as per **(a)**).

We consider an execution of  $\Pi$  with inputs  $(x_1, \dots, x_7)$  where  $x_i$  denotes the input of  $P_i$  and analyze four different scenarios. Before describing the scenarios, we define some useful notation. In Rounds 1 and 2, let  $\text{pc}_{i \rightarrow j}^r$ , where  $r \in [2], \{i, j\} \in [7]$  denote the pairwise-private communication from  $P_i$  to  $P_j$  in Round  $r$ , as per an execution where everyone behaves honestly. Next, we use  $\widetilde{\text{pc}}_{i \rightarrow j}^2$  to denote the messages that  $P_i$  ( $i \in [7]$ ) is supposed to send in Round 2 to  $P_j$  ( $j \in [6] \setminus i$ ) in case  $P_i$  did not receive Round 1 message from  $P_1$ . Note that this communication could be potentially different from what  $P_i$  would send in an honest execution. Lastly, since Round 3 messages to  $Q$  could potentially be different for each of the four scenarios, we index them additionally with  $\ell$  indicating the scenario i.e  $\text{pc}_{j \rightarrow 7}^{3, \ell}$  denotes  $P_j$ 's Round 3 message to  $Q$  in Scenario  $\ell$  ( $j \in [6], \ell \in [4]$ ). These messages may be a function of the common reference string (denoted by  $\text{crs}$ ) and the PKI setup. Let  $\alpha_i$  denote the output of the PKI setup to party  $P_i$ . A party's view comprises of  $\text{crs}$ ,  $\alpha_i$ , its input, randomness and incoming messages.

Due to the involved nature of the scenarios, we begin with an intuitive description. Broadly speaking, this argument involves partitioning the parties  $\{P_1, \dots, P_6\}$  into two sets  $\{P_1, P_2, P_6\}$  and  $\{P_3, P_4, P_5\}$ . Looking ahead, the final scenario (corresponding to the final contradiction) is designed in a manner that allows a corrupt  $Q$  to obtain: (i) output with respect to some input of  $P_1$  using the communication from  $\{P_1, P_2, P_6\}$  and (ii) output with respect to a different input of  $P_1$  using the communication from  $\{P_3, P_4, P_5\}$ . Tracing back, we carefully design the other scenarios which let us make the following crucial inferences - Scenario 1 and 2 let us conclude that if  $P_1$  behaves honestly only in its messages to  $P_6$ , then the communication from  $\{P_3, P_4, P_5\}$  to  $Q$  in such a case must enable  $Q$  to obtain output with respect to some  $x_1^*$ , which is independent of  $x_1$ , with non-negligible probability. On the other hand, Scenario 3 involves corrupt  $\{P_3, P_4, P_5\}$  who pretend to have received no message from  $P_1$ ; which lets us conclude that the messages from  $\{P_1, P_2, P_6\}$  in such a case must enable  $Q$  to obtain output with respect to honest input  $x_1$  of  $P_1$ . Combining the above two inferences in the final scenario lets us reach the final contradiction.

Following is a description of the scenarios. In each of these scenarios, we assume that the adversary uses the honest input on behalf of the corrupt parties and its malicious behaviour is limited to dropping some of the messages that were received or supposed to be sent by the actively corrupt parties. The views of the parties across various scenarios are described in Tables 11 – 14.

**Scenario 1:** Adversary corrupts  $\{P_1, P_6\}$ .  $P_1$  does not communicate throughout the protocol.  $P_6$  behaves honestly in Round 1 and Round 2 (thereby would send  $\widetilde{\text{pc}}_{6 \rightarrow j}^2$  for  $j \in [5]$ ) and aborts (does not communicate) in Round 3.

**Scenario 2:** Adversary corrupts  $\{P_1, P_6\}$ .  $P_1$  does not communicate throughout the protocol.  $P_6$  behaves honestly in Round 1 and Round 2, except that  $P_6$  pretends to have received Round 1 message from  $P_1$  (thereby would send  $\widetilde{\text{pc}}_{6 \rightarrow j}^2$  for  $j \in [5]$ ). Note that it is possible for  $P_6$  to pretend in such a manner as adversary corrupts both  $P_1, P_6$ . Lastly,  $P_6$  aborts in Round 3.

**Scenario 3:** Adversary corrupts  $\{P_3, P_4, P_5\}$ . All corrupt parties behave honestly in Round 1. In Round 2,  $\{P_3, P_4, P_5\}$  only communicate towards  $P_6$ , towards whom they pretend that they did not receive Round 1 message from  $P_1$  (i.e  $P_i$  sends  $\widetilde{\text{pc}}_{i \rightarrow 6}^2$  to  $P_6$  for  $i \in \{3, 4, 5\}$ ). Lastly,  $\{P_3, P_4, P_5\}$  abort in Round 3.

**Scenario 4:** Adversary corrupts  $\{P_1, P_2, Q\}$  who do the following:<sup>7</sup>

Round 1:  $P_1$  behaves honestly only to  $\{P_2, P_6, Q\}$  (only  $P_6$  among the honest parties).  $P_2$  and  $Q$  behave honestly.

Round 2:  $P_1$  behaves honestly only to  $\{P_2, P_6, Q\}$ .  $P_2$  and  $Q$  pretend towards  $\{P_3, P_4, P_5\}$  as if they did not receive Round 1 message from  $P_1$  (i.e send  $\widetilde{\text{pc}}_{i \rightarrow j}^2$  to  $P_j$  for  $i \in \{2, 7\}$ ,  $j \in \{3, 4, 5\}$ ). Towards  $\{P_1, P_2, P_6\}$  (only  $P_6$  among honest parties),  $P_2$  and  $Q$  act as if Round 1 message had been received from  $P_1$  (i.e send  $\text{pc}_{i \rightarrow j}^2$  to  $P_j$  for  $i \in \{2, 7\}$ ,  $j \in \{1, 2, 6\} \setminus i$ ).

Round 3:  $P_1$  and  $P_2$  drop the Round 2 messages obtained from  $\{P_3, P_4, P_5\}$  (to emulate Scenario 3) and communicate to  $Q$  accordingly.

	View <sub>1</sub>	View <sub>2</sub>	View <sub>3</sub>	View <sub>4</sub>	View <sub>5</sub>	View <sub>6</sub>	View <sub>7</sub>
Initial Input	$(x_1, r_1, \text{crs}, \alpha_1)$	$(x_2, r_2, \text{crs}, \alpha_2)$	$(x_3, r_3, \text{crs}, \alpha_3)$	$(x_4, r_4, \text{crs}, \alpha_4)$	$(x_5, r_5, \text{crs}, \alpha_5)$	$(x_6, r_6, \text{crs}, \alpha_6)$	$(x_7, r_7, \text{crs}, \alpha_7)$
Round 1	$\{\text{pc}_{j \rightarrow 1}^1\}_{j \in [7] \setminus \{1\}}$	$\{\text{pc}_{j \rightarrow 2}^1\}_{j \in [7] \setminus \{1, 2\}}$	$\{\text{pc}_{j \rightarrow 3}^1\}_{j \in [7] \setminus \{1, 3\}}$	$\{\text{pc}_{j \rightarrow 4}^1\}_{j \in [7] \setminus \{1, 4\}}$	$\{\text{pc}_{j \rightarrow 5}^1\}_{j \in [7] \setminus \{1, 5\}}$	$\{\text{pc}_{j \rightarrow 6}^1\}_{j \in [7] \setminus \{1, 6\}}$	$\{\text{pc}_{j \rightarrow 7}^1\}_{j \in [7] \setminus \{1, 7\}}$
Round 2	$\{\widetilde{\text{pc}}_{j \rightarrow 1}^2\}_{j \in [7] \setminus \{1\}}$	$\{\widetilde{\text{pc}}_{j \rightarrow 2}^2\}_{j \in [7] \setminus \{1, 2\}}$	$\{\widetilde{\text{pc}}_{j \rightarrow 3}^2\}_{j \in [7] \setminus \{1, 3\}}$	$\{\widetilde{\text{pc}}_{j \rightarrow 4}^2\}_{j \in [7] \setminus \{1, 4\}}$	$\{\widetilde{\text{pc}}_{j \rightarrow 5}^2\}_{j \in [7] \setminus \{1, 5\}}$	$\{\widetilde{\text{pc}}_{j \rightarrow 6}^2\}_{j \in [7] \setminus \{1, 6\}}$	–
Round 3	–	–	–	–	–	–	$\{\text{pc}_{j \rightarrow 7}^{3,1}\}_{j \in \{2, 3, 4, 5\}}$

Table 11: Views of  $\{P_1 \dots P_7\}$  in Scenario 1.

	View <sub>1</sub>	View <sub>2</sub>	View <sub>3</sub>	View <sub>4</sub>	View <sub>5</sub>	View <sub>6</sub>	View <sub>7</sub>
Initial Input	$(x_1, r_1, \text{crs}, \alpha_1)$	$(x_2, r_2, \text{crs}, \alpha_2)$	$(x_3, r_3, \text{crs}, \alpha_3)$	$(x_4, r_4, \text{crs}, \alpha_4)$	$(x_5, r_5, \text{crs}, \alpha_5)$	$(x_6, r_6, \text{crs}, \alpha_6)$	$(x_7, r_7, \text{crs}, \alpha_7)$
Round 1	$\{\text{pc}_{j \rightarrow 1}^1\}_{j \in [7] \setminus \{1\}}$	$\{\text{pc}_{j \rightarrow 2}^1\}_{j \in [7] \setminus \{1, 2\}}$	$\{\text{pc}_{j \rightarrow 3}^1\}_{j \in [7] \setminus \{1, 3\}}$	$\{\text{pc}_{j \rightarrow 4}^1\}_{j \in [7] \setminus \{1, 4\}}$	$\{\text{pc}_{j \rightarrow 5}^1\}_{j \in [7] \setminus \{1, 5\}}$	$\{\text{pc}_{j \rightarrow 6}^1\}_{j \in [7] \setminus \{1, 6\}}$	$\{\text{pc}_{j \rightarrow 7}^1\}_{j \in [7] \setminus \{1, 7\}}$
Round 2	$\{\widetilde{\text{pc}}_{j \rightarrow 1}^2\}_{j \in \{2, 3, 4, 5, 7\}}$ $\text{pc}_{6 \rightarrow 1}^2$	$\{\widetilde{\text{pc}}_{j \rightarrow 2}^2\}_{j \in \{3, 4, 5, 7\}}$ $\text{pc}_{6 \rightarrow 2}^2$	$\{\widetilde{\text{pc}}_{j \rightarrow 3}^2\}_{j \in \{2, 4, 5, 7\}}$ $\text{pc}_{6 \rightarrow 3}^2$	$\{\widetilde{\text{pc}}_{j \rightarrow 4}^2\}_{j \in \{2, 3, 5, 7\}}$ $\text{pc}_{6 \rightarrow 4}^2$	$\{\widetilde{\text{pc}}_{j \rightarrow 5}^2\}_{j \in \{2, 3, 4, 7\}}$ $\text{pc}_{6 \rightarrow 5}^2$	$\{\widetilde{\text{pc}}_{j \rightarrow 6}^2\}_{j \in \{2, 3, 4, 5, 7\}}$	–
Round 3	–	–	–	–	–	–	$\{\text{pc}_{j \rightarrow 7}^{3,2}\}_{j \in \{2, 3, 4, 5\}}$

Table 12: Views of  $\{P_1 \dots P_7\}$  in Scenario 2.

<sup>7</sup>Generally, communication between corrupt parties need not be specified but we include it here for easier understanding of Table 14.

	View <sub>1</sub>	View <sub>2</sub>	View <sub>3</sub>	View <sub>4</sub>	View <sub>5</sub>	View <sub>6</sub>	View <sub>7</sub>
Initial Input	$(x_1, r_1, \text{crs}, \alpha_1)$	$(x_2, r_2, \text{crs}, \alpha_2)$	$(x_3, r_3, \text{crs}, \alpha_3)$	$(x_4, r_4, \text{crs}, \alpha_4)$	$(x_5, r_5, \text{crs}, \alpha_5)$	$(x_6, r_6, \text{crs}, \alpha_6)$	$(x_7, r_7, \text{crs}, \alpha_7)$
Round 1	$\{\widehat{\text{pc}}_{j \rightarrow 1}^1\}_{j \in [7] \setminus \{1\}}$	$\{\widehat{\text{pc}}_{j \rightarrow 2}^1\}_{j \in [7] \setminus \{2\}}$	$\{\widehat{\text{pc}}_{j \rightarrow 3}^1\}_{j \in [7] \setminus \{3\}}$	$\{\widehat{\text{pc}}_{j \rightarrow 4}^1\}_{j \in [7] \setminus \{4\}}$	$\{\widehat{\text{pc}}_{j \rightarrow 5}^1\}_{j \in [7] \setminus \{5\}}$	$\{\widehat{\text{pc}}_{j \rightarrow 6}^1\}_{j \in [7] \setminus \{6\}}$	$\{\widehat{\text{pc}}_{j \rightarrow 7}^1\}_{j \in [7] \setminus \{7\}}$
Round 2	$\{\widehat{\text{pc}}_{j \rightarrow 1}^2\}_{j \in \{2,6,7\}}$	$\{\widehat{\text{pc}}_{j \rightarrow 2}^2\}_{j \in \{1,6,7\}}$	$\{\widehat{\text{pc}}_{j \rightarrow 3}^2\}_{j \in \{1,2,6,7\}}$	$\{\widehat{\text{pc}}_{j \rightarrow 4}^2\}_{j \in \{1,2,6,7\}}$	$\{\widehat{\text{pc}}_{j \rightarrow 5}^2\}_{j \in \{1,2,6,7\}}$	$\{\widehat{\text{pc}}_{j \rightarrow 6}^2\}_{j \in \{1,2,7\}}$ $\{\widehat{\text{pc}}_{j \rightarrow 6}^2\}_{j \in \{3,4,5\}}$	–
Round 3	–	–	–	–	–	–	$\{\widehat{\text{pc}}_{j \rightarrow 7}^{3,3}\}_{j \in \{1,2,6\}}$

Table 13: Views of  $\{P_1 \dots P_7\}$  in Scenario 3.

	View <sub>1</sub>	View <sub>2</sub>	View <sub>3</sub>	View <sub>4</sub>	View <sub>5</sub>	View <sub>6</sub>	View <sub>7</sub>
Initial Input	$(x_1, r_1, \text{crs}, \alpha_1)$	$(x_2, r_2, \text{crs}, \alpha_2)$	$(x_3, r_3, \text{crs}, \alpha_3)$	$(x_4, r_4, \text{crs}, \alpha_4)$	$(x_5, r_5, \text{crs}, \alpha_5)$	$(x_6, r_6, \text{crs}, \alpha_6)$	$(x_7, r_7, \text{crs}, \alpha_7)$
Round 1	$\{\widehat{\text{pc}}_{j \rightarrow 1}^1\}_{j \in [7] \setminus \{1\}}$	$\{\widehat{\text{pc}}_{j \rightarrow 2}^1\}_{j \in [7] \setminus \{2\}}$	$\{\widehat{\text{pc}}_{j \rightarrow 3}^1\}_{j \in [7] \setminus \{1,3\}}$	$\{\widehat{\text{pc}}_{j \rightarrow 4}^1\}_{j \in [7] \setminus \{1,4\}}$	$\{\widehat{\text{pc}}_{j \rightarrow 5}^1\}_{j \in [7] \setminus \{1,5\}}$	$\{\widehat{\text{pc}}_{j \rightarrow 6}^1\}_{j \in [7] \setminus \{6\}}$	$\{\widehat{\text{pc}}_{j \rightarrow 7}^1\}_{j \in [7] \setminus \{7\}}$
Round 2	$\{\widehat{\text{pc}}_{j \rightarrow 1}^2\}_{j \in \{3,4,5\}}$ $\{\widehat{\text{pc}}_{j \rightarrow 1}^2\}_{j \in \{2,6,7\}}$	$\{\widehat{\text{pc}}_{j \rightarrow 2}^2\}_{j \in \{3,4,5\}}$ $\{\widehat{\text{pc}}_{j \rightarrow 2}^2\}_{j \in \{1,6,7\}}$	$\{\widehat{\text{pc}}_{j \rightarrow 3}^2\}_{j \in \{2,4,5,7\}}$ $\widehat{\text{pc}}_{6 \rightarrow 3}^2$	$\{\widehat{\text{pc}}_{j \rightarrow 4}^2\}_{j \in \{2,3,5,7\}}$ $\widehat{\text{pc}}_{6 \rightarrow 4}^2$	$\{\widehat{\text{pc}}_{j \rightarrow 5}^2\}_{\{2,3,4,7\}}$ $\widehat{\text{pc}}_{6 \rightarrow 5}^2$	$\{\widehat{\text{pc}}_{j \rightarrow 6}^2\}_{j \in \{3,4,5\}}$ $\{\widehat{\text{pc}}_{j \rightarrow 6}^2\}_{j \in \{1,2,7\}}$	–
Round 3	–	–	–	–	–	–	$\{\widehat{\text{pc}}_{j \rightarrow 7}^{3,4} \equiv \widehat{\text{pc}}_{j \rightarrow 7}^{3,3}\}_{j \in \{1,2,6\}}$ $\{\widehat{\text{pc}}_{j \rightarrow 7}^{3,4} \equiv \widehat{\text{pc}}_{j \rightarrow 6}^{3,2}\}_{j \in \{3,4,5\}}$

Table 14: Views of  $\{P_1 \dots P_7\}$  in Scenario 4.

The proof outline is as follows. First, we show that  $\Pi$  must be such that if Scenario 1 occurs, then the output obtained by  $Q$  is computed on some input  $x_1^*$  that is independent of  $x_1$  with non-negligible probability. Next, we show this is also the case for Scenario 2. Since this inference may appear counter-intuitive, we elaborate the argument in some detail below. Note that the difference between Scenario 1 and 2 lies in the communication from  $P_6$  to honest parties  $\{P_2, P_3, P_4, P_5\}$  in Round 2. While in the former,  $P_6$  acts as if he did not receive Round 1 message from  $P_1$ ; in the latter he pretends as if he did receive Round 1 message from  $P_1$ . To prove that  $Q$  obtains an output on  $x_1^*$  that is independent of  $x_1$  with non-negligible probability, we define a sequence of hybrids  $\text{hyb}_0, \dots, \text{hyb}_4$ . Specifically,  $\text{hyb}_0$  and  $\text{hyb}_4$  refer to Scenario 1 and 2 respectively and  $\text{hyb}_i$  is same as  $\text{hyb}_{i-1}$  ( $i \in \{1, \dots, 4\}$ ) except that  $P_6$  acts towards  $P_{i+1}$  that he did receive Round 1 message from  $P_1$ . We show that in each hybrid, the output obtained by  $Q$  is w.r.t.  $x_1^*$  with non-negligible probability. Next, if Scenario 3 occurs, then the output obtained by  $Q$  must be computed on  $x_1$  (honest input of  $P_1$ ) due to correctness of  $\Pi$ . Lastly, we show that such a protocol  $\Pi$  is susceptible to an attack by  $\{P_1, P_2, Q\}$  which allows  $Q$  to obtain both the above evaluations of  $f$  (i.e., on both  $x_1$  and  $x_1^*$  of  $P_1$ ), which is a contradiction to security of  $\Pi$ . We now prove a sequence of lemmas to complete our proof.

**Lemma 5.2.**  $\Pi$  must be such that if Scenario 1 occurs, then there exists  $x_1^* \in \{0, 1\}$  such that  $Q$  obtains an output computed on  $x_1^*$  and the honest parties' inputs with non-negligible probability.

*Proof.* It follows from the **god** and correctness properties of  $\Pi$  that an honest  $Q$  must receive an output that is computed on the honest parties' inputs with overwhelming probability. Next, the output obtained by  $Q$  should be computed on some  $x_1'$  that is independent of  $x_1$ , as the communication throughout the protocol is independent of  $x_1$ . Since there are only two possible values for  $x_1'$ , there must exist  $x_1^* \in \{0, 1\}$  such that the output is computed on  $x_1^*$  with non-negligible probability.  $\square$

**Lemma 5.3.**  $\Pi$  must be such that if Scenario 2 occurs, then there exists  $x_1^* \in \{0, 1\}$  such that  $Q$  obtains an output computed on  $x_1^*$  and the honest parties' inputs with non-negligible probability.

*Proof.* First, it follows from the **god** and correctness properties that  $Q$  must receive an output that is computed on the honest parties' inputs with overwhelming probability. Next, we note that the difference between Scenario 1 and 2 lies in the communication from  $P_6$  to honest parties  $\{P_2, P_3, P_4, P_5\}$  in Round 2. While in the former,  $P_6$  acts as if he did not receive Round 1 message from  $P_1$  (sends  $\widetilde{\text{pc}}_{6 \rightarrow j}^2$  for  $j \in \{2, 3, 4, 5\}$ ); in the latter he pretends as if he did receive Round 1 message from  $P_1$  (sends  $\text{pc}_{6 \rightarrow j}^2$  for  $j \in \{2, 3, 4, 5\}$ ). We prove the statement is true via the following sequence of hybrids:

**hyb<sub>0</sub>**: Same as Scenario 1.

Recall that the statement is true in Scenario 1 (**hyb<sub>0</sub>**) by Lemma 5.2. In particular, there exists  $x_1^* \in \{0, 1\}$  such that  $Q$  obtains an output computed on  $x_1^*$  and the honest parties' inputs with non-negligible probability. We consider this special  $x_1^*$  in the subsequent hybrids.

**hyb<sub>0,1</sub>**: Same as **hyb<sub>0</sub>** except that  $P_2$  is also corrupted by the adversary, who follows the protocol in the first two rounds and aborts in Round 3.

Assume for contradiction that the statement is false in this hybrid. That is,  $Q$  obtains an output computed on  $x_1^*$  with negligible probability. Since  $x_1$  only has two values  $\{0, 1\}$  and an honest  $Q$  must receive an output by the **god** property, the output learned by  $Q$  must be computed on  $\neg x_1^*$  with overwhelming probability. We focus on the party  $P_3$  who is honest in both **hyb<sub>0</sub>** and **hyb<sub>0,1</sub>**. Then  $Q$  obtains an output computed on  $(x_1^*, x_3)$  with non-negligible probability in **hyb<sub>0</sub>**, and obtains an output computed on  $(\neg x_1^*, x_3)$  with overwhelming probability in **hyb<sub>0,1</sub>**.

Now consider another scenario where the adversary corrupts  $\{P_1, P_6, Q\}$ , who behave in the same way as in **hyb<sub>0</sub>**. In this scenario,  $Q$  can learn the output in **hyb<sub>0</sub>** as well as in **hyb<sub>0,1</sub>** (by dropping  $P_2$ 's Round 3 message). In other words,  $Q$  can learn the output computed on both  $(x_1^*, x_3)$  and  $(\neg x_1^*, x_3)$  with non-negligible probability, namely both  $x_3^0$  and  $x_3^1$ , contradicting the security of  $\Pi$ .

**hyb<sub>0,2</sub>**: Same as **hyb<sub>0,1</sub>** but  $P_6$  sends  $\text{pc}_{6 \rightarrow 2}^2$  to  $P_2$  (as opposed to  $\widetilde{\text{pc}}_{6 \rightarrow 2}^2$ ).

The only difference between **hyb<sub>0,1</sub>** and **hyb<sub>0,2</sub>** is the Round 2 message sent from  $P_6$  to  $P_2$ . Note that both parties are corrupted and the communication between them is mentioned only for clarity. Since  $P_2$  aborts in Round 3,  $Q$ 's view in this hybrid is identical to **hyb<sub>0,1</sub>**, hence the statement remains true.

**hyb<sub>1</sub>**: Same as **hyb<sub>0</sub>** except that  $P_6$  sends  $\text{pc}_{6 \rightarrow 2}^2$  to  $P_2$  (as opposed to  $\widetilde{\text{pc}}_{6 \rightarrow 2}^2$ ).

Notice that the only difference between **hyb<sub>0,2</sub>** and **hyb<sub>1</sub>** is whether  $P_2$  aborts in Round 3 or not. We can argue the statement remains true in this hybrid similarly as in the argument for **hyb<sub>0,1</sub>**.

**hyb<sub>2</sub>**: Same as **hyb<sub>1</sub>**, except that  $P_6$  sends  $\text{pc}_{6 \rightarrow 3}^2$  to  $P_3$  (as opposed to  $\widetilde{\text{pc}}_{6 \rightarrow 3}^2$ ).

We can argue the statement remains true in the same way as from **hyb<sub>0</sub>** to **hyb<sub>1</sub>** via a sequence of hybrids. The only difference is that we need to consider  $P_4$  as the common honest party (instead of  $P_3$ ).

hyb<sub>3</sub>: Same as hyb<sub>2</sub> except that  $P_6$  sends  $\text{pc}_{6 \rightarrow 4}^2$  to  $P_4$  (as opposed to  $\widetilde{\text{pc}}_{6 \rightarrow 4}^2$ ).

We can argue the statement remains true in the same way as from hyb<sub>0</sub> to hyb<sub>1</sub> via a sequence of hybrids (where we consider  $P_3$  as the common honest party).

hyb<sub>4</sub>: Same as hyb<sub>3</sub> except that  $P_6$  sends  $\text{pc}_{6 \rightarrow 5}^2$  to  $P_5$  (as opposed to  $\widetilde{\text{pc}}_{6 \rightarrow 5}^2$ ). Note that this is the same as Scenario 2.

We can argue the statement remains true in the same way as from hyb<sub>0</sub> to hyb<sub>1</sub> via a sequence of hybrids (where we consider  $P_3$  as the common honest party). This concludes the proof.  $\square$

**Lemma 5.4.**  *$\Pi$  must be such that if Scenario 3 occurs, then the output obtained by  $Q$  must be computed on the honest parties' inputs  $(x_1, x_2, x_6, x_7)$  with overwhelming probability.*

*Proof.* This follows directly from the **god** and correctness properties of  $\Pi$ .  $\square$

**Claim 5.5.**  *$\Pi$  is such that the view of  $\{P_3, P_4, P_5\}$  in Scenario 4 is identically distributed to their respective views in Scenario 2.*

*Proof.* Consider the view of  $P_i$  ( $i \in \{3, 4, 5\}$ ). In both Scenario 2 and Scenario 4,  $P_i$  does not receive communication from  $P_1$  in Round 1 and Round 2, receives  $\text{pc}_{6 \rightarrow i}^2$  from  $P_6$  and  $\widetilde{\text{pc}}_{j \rightarrow i}^2$  from  $j \in [7] \setminus \{1, 6, i\}$  (refer to Tables 12, 14). Thereby, the claim follows.  $\square$

**Claim 5.6.**  *$\Pi$  is such that the view of  $\{P_1, P_2, P_6\}$  in Scenario 4 is identically distributed (or subsumes) their respective views in Scenario 3.*

*Proof.* Consider the view of  $P_6$ . In both Scenario 3 and Scenario 4, honest  $P_6$  receives communication from  $P_1$  in Round 1 and Round 2, receives  $\text{pc}_{j \rightarrow 6}^2$  for  $j \in \{1, 2, 7\}$  and receives  $\widetilde{\text{pc}}_{j \rightarrow 6}^2$  from  $j \in \{3, 4, 5\}$ . Thereby, the claim holds w.r.t  $P_6$ . Next, suppose the corrupt parties  $\{\widetilde{P}_1, P_2\}$  in Scenario 4 discard Round 2 messages from  $\{P_3, P_4, P_5\}$ . Consider this updated view of  $P_1$  which would constitute  $\text{pc}_{j \rightarrow 1}^2$  for  $j \in \{2, 6, 7\}$  (in addition to Round 1 messages). It is easy to check that this is identically distributed to view of honest  $P_1$  in Scenario 3. Similar argument can be made w.r.t  $P_2$  (refer Tables 13, 14). Thus, the claim holds.  $\square$

Now we prove the final contradiction. Consider the special  $x_1^*$  in Lemma 5.2. Suppose Scenario 4 occurs for  $(x_1, x_3, x_4, x_6)$  where  $x_1 = \neg x_1^*$ . We claim that  $Q$  must obtain the output computed w.r.t  $x_1$  with overwhelming probability. Let  $Q$  locally update his view in Scenario 4 to discard the Round 3 messages from  $\{P_3, P_4, P_5\}$ . This updated view is identically distributed to the view of an honest  $Q$  in Scenario 3. This holds since the Round 3 messages from  $\{P_1, P_2, P_6\}$  are identically distributed to those received in Scenario 3 (can be inferred from Claim 5.6). We can thus conclude that  $Q$  can carry out output computation similar to honest  $Q$  in Scenario 3 to obtain an output computed on  $(\neg x_1^*, x_6)$  with overwhelming probability (Lemma 5.4).

Next, we claim that  $Q$  can obtain the output computed on  $x_1^*$  with non-negligible probability as well. Consider a scenario related to Scenario 2 where  $P_2$  additionally aborts in Round 3. It follows from the proof of hyb<sub>0,1</sub> in Lemma 5.3 that  $Q$  can learn an output computed on  $x_1^*$  and the honest parties' inputs with non-negligible probability in this scenario. Next, let  $Q$  locally update his view in Scenario 4 by discarding the Round 3 messages from  $\{P_1, P_2, P_6\}$ . This updated view

is identically distributed to the view of an honest  $Q$  in the scenario mentioned above (related to Scenario 2). This holds since Round 3 messages of  $\{P_3, P_4, P_5\}$  are identically distributed to those received by honest  $Q$  in Scenario 2 (can be inferred from [Claim 5.5](#)). Therefore,  $Q$  can learn an output computed on  $(x_1^*, x_3, x_4)$ .

To conclude,  $\Pi$  is such that an adversary corrupting  $\{P_1, P_2, Q\}$  can obtain an output computed on  $(-x_1^*, x_6)$  as well as an output computed on  $(x_1^*, x_3, x_4)$  with non-negligible probability, for any  $(x_3, x_4, x_6)$ . This contradicts the security of  $\Pi$  as  $Q$  cannot learn both outputs as per the definition of  $f$ ; completing the proof of [Theorem 5.1](#) for the setting  $n = 7, t = 3$ .

Lastly, we show how the above proof can be extended for any  $3 \lceil n/7 \rceil \leq t < n/2$  using party partitioning technique. Assume towards a contradiction, that there exists a three-round  $n$ -party solitary MPC  $\Pi'$  that achieves `god` against  $t$  corruptions where  $2t < n \leq 7 \lfloor t/3 \rfloor$  (equivalent to  $3 \lceil n/7 \rceil \leq t < n/2$ <sup>8</sup>) and  $t \geq 3$ . Then,  $\Pi'$  can be transformed to a three-round 7-party solitary MPC protocol  $\Pi$  that achieves `god` against three corruptions as follows: Partition the set of  $n$  parties into 7 disjoint groups, say  $S_i$  ( $i \in [7]$ ), where  $S_1, \dots, S_6$  are each of size  $\lfloor t/3 \rfloor$  and  $S_7$  comprises of the remaining  $(n - 6 \lfloor t/3 \rfloor)$  parties. Let  $P_i$  ( $i \in [7]$ ) in  $\Pi$  emulate the steps of parties in  $S_i$  during  $\Pi'$ . It is easy to see that security of  $\Pi'$  implies security of  $\Pi$  (as corruption of upto 3 parties in  $\Pi$  is analogous to corruption of upto  $t$  parties in  $\Pi'$ ). However, our proof argument above showed the impossibility of such a three-round 7-party solitary MPC protocol  $\Pi$  that achieves `god` against three corruptions. We have thus arrived at a contradiction, completing the proof of [Theorem 5.1](#). Note that for certain cases, such as  $n = 8$ , this range of values of  $(n, t)$  is not meaningful. However, this generalization is relevant for sufficiently large values of  $n$ .  $\square$

Before concluding the section, we briefly discuss why our proof approach of [Theorem 5.1](#) breaks down when  $t = 2$  (circumventing the lower bound when  $t = 1$  is already demonstrated by the upper bounds of [Section 6.2](#)). Suppose the scenarios above are extended in a natural manner to a five party setting  $\{P_1, P_2, P_3, P_4, P_5 = Q\}$  with two corruptions where the partitions comprise of  $\{P_1, P_4\}$  and  $\{P_2, P_3\}$  (analogous to  $\{P_1, P_2, P_6\}$  and  $\{P_3, P_4, P_5\}$  in the above argument). We observe that while the inferences corresponding to Scenario 1 and 3 still hold, the proof of [Lemma 5.3](#) breaks down as the argument involving the hybrids does not work in the case of two corruptions. This is because the scenarios corresponding to the hybrids would already involve corruptions of  $\{P_1, P_4\}$  (analogous to  $\{P_1, P_6\}$  in the general argument) and demand an additional corruption (i.e the party whose message changed across the hybrids) which is not possible when  $t = 2$ . Therefore, we cannot conclude that when Scenario 2 occurs, the output is computed on some input  $x_1^*$ , that is independent of  $x_1$ , with non-negligible probability.

We believe the above insight may be useful in potentially designing a three-round upper bound for the case of  $t = 2$  corruptions in future. We leave open the question of designing a three-round solitary MPC or alternately proving its impossibility for the case of  $t = 2$  corruptions.

## 5.2 General Five-Round Protocol

In this section, we present a five-round solitary output MPC protocol with guaranteed output delivery that works for any  $n$  in the presence of an honest majority - that is, any  $t < n/2$  where  $n$  is the number of parties and  $t$  is the number of corrupt parties. Our protocol uses the following primitives: a  $(\frac{n}{2} + 1)$ -out-of- $n$  decentralized threshold FHE scheme  $\text{dTFHE} = (\text{dTFHE}.\text{DistGen},$

---

<sup>8</sup>Note that  $n \leq 7 \lfloor t/3 \rfloor \iff n/7 \leq \lfloor t/3 \rfloor \iff \lceil n/7 \rceil \leq t/3 \iff 3 \lceil n/7 \rceil \leq t$ .

dTFHE.Enc, dTFHE.PartialDec, dTFHE.Eval, dTFHE.Combine), a digital signature scheme (Gen, Sign, Verify), and a simulation-extractible NIZK argument (NIZK.Setup, NIZK.Prove, NIZK.Verify). We use the NIZK argument for two NP languages  $L_1, L_2$  defined in [Section 3.3](#). All of them can be built assuming LWE [[BGG<sup>+</sup>18](#), [CCH<sup>+</sup>19](#), [PS19](#)]. Formally, we show the following theorem:

**Theorem 5.7.** *Assuming LWE, protocol  $\Pi_{5\text{-round}}$  described below is a five-round secure solitary output MPC protocol with **god** with a PKI setup and pairwise-private channels. The protocol works for any  $n$ , any function and is secure against a malicious rushing adversary that can corrupt any  $t < n/2$  parties.*

**Overview.** Consider  $n$  parties  $P_1, \dots, P_n$  who wish to evaluate function  $f : (\{0, 1\}^\lambda)^{n-1} \rightarrow \{0, 1\}^\lambda$ . We also denote  $P_n$  as the output receiving party  $Q$ . In some places, we use the notation  $\text{msg}^{i \rightarrow j}$  to indicate that the message was sent by party  $P_i$  to  $P_j$ . At a high level, our protocol works as follows. In Round 1, each party  $P_i$  sends to every other party a dTFHE encryption  $\llbracket x_i \rrbracket$  along with a NIZK argument  $\pi_i$  proving that the encryption is well formed. On top of that,  $P_i$  also attaches its signature  $\sigma_i \leftarrow \text{Sign}(\text{skey}_i, (\llbracket x_i \rrbracket, \pi_i))$ . In Round 2, each party sends all the messages it received in Round 1 to  $Q$ . In Round 3,  $Q$  first initializes a string  $\text{msg} = \perp$  and does the following for each  $i \in [n]$ : if it received a *valid* message from  $P_i$  in Round 1, (where *valid* means the signature  $\sigma_i$  and the NIZK  $\pi_i$  verifies successfully) it includes the message in  $\text{msg}$  and sets a value  $\text{ct}_i = \llbracket x_i \rrbracket$ . Else, in Round 2, if a different party  $P_{i_1}$ , forwards a *valid* message  $(\llbracket x_i \rrbracket^{i_1 \rightarrow n}, \pi^{i_1 \rightarrow n}, \sigma^{i_1 \rightarrow n})$  received from  $P_i$  in Round 1, include that in  $\text{msg}$  and set  $\text{ct}_i$  to be  $\llbracket x_i \rrbracket^{i_1 \rightarrow n}$ . If no such  $i_1$  exists, set  $\text{ct}_i = \perp$  and append  $\perp$  to  $\text{msg}$ . Then,  $Q$  sends  $\text{msg}$  and a signature on it  $\sigma_{\text{msg}}$  to all parties. In Round 4, each party sends the tuple received from  $Q$  in Round 3 to every other party. Finally, in Round 5, each party  $P_i$  sends its partial decryption (along with a NIZK) on the homomorphically evaluated ciphertext  $\llbracket y \rrbracket = \text{dTFHE.Eval}(f, \text{ct}_1, \dots, \text{ct}_n)$  if: (i) in Round 3,  $Q$  sent  $(\text{msg}, \sigma_{\text{msg}})$  such that  $\sigma_{\text{msg}}$  verifies, (ii) it did not receive a different tuple  $(\text{msg}', \sigma_{\text{msg}'})$  from another party in Round 4 such that  $\sigma_{\text{msg}'}$  verifies, (iii) In the string  $\text{msg}$ , every tuple of the form  $(\llbracket x_j \rrbracket, \pi_j, \sigma_j)$  is *valid*, (iv) for every party  $P_k$ , if  $P_i$  received a *valid* message from  $P_k$  in Round 1, then in  $Q$ 's Round 3 message  $\text{msg}$ , there must exist *some valid* tuple of the form  $(\llbracket x'_k \rrbracket, \pi'_k, \sigma'_k)$  on behalf of  $P_k$  (not necessarily the one  $P_i$  received in Round 1). After Round 5,  $Q$  combines all the partial decryptions (if the NIZK verifies) to recover the output. Our protocol is formally described below. We defer the security proof to [Appendix C](#).

**CRS:** Send  $\text{crs} \leftarrow \text{NIZK.Setup}(1^\lambda)$  to every party.

**PKI Setup:**

- For each  $i \in [n]$ : sample  $(\text{pk}_i, \text{sk}_i) \leftarrow \text{dTFHE.DistGen}(1^\lambda, 1^d, i; r_i)$  and  $(\text{vkey}_i, \text{skey}_i) \leftarrow \text{Gen}(1^\lambda)$ .
- Public key:  $\text{pk} = \text{pk}_1 \parallel \dots \parallel \text{pk}_n$  and  $\{\text{vkey}_i\}_{i \in [n]}$ .
- Secret keys:  $(\text{sk}_i, r_i, \text{skey}_i)$  to party  $P_i$  for each  $i \in [n]$ .

**Inputs:** For each  $i \in [n]$ , party  $P_i$  has an input  $x_i \in \{0, 1\}^\lambda$ .

**Protocol:**

1. **Round 1:** For each  $i \in [n]$ :

- $P_i$  computes  $\llbracket x_i \rrbracket \leftarrow \text{dTFHE.Enc}(\text{pk}, x_i; \rho_i)$  using randomness  $\rho_i$ ,  $\pi_i \leftarrow \text{NIZK.Prove}(\text{crs}, \text{st}_i, \text{wit}_i)$  for  $\text{st}_i \in L_1$  where  $\text{st}_i = (\llbracket x_i \rrbracket, \text{pk})$  and  $\text{wit}_i = (x_i, \rho_i)$ .

- Then, compute  $\sigma_i \leftarrow \text{Sign}(\text{skey}_i, (\llbracket x_i \rrbracket, \pi_i))$  and send  $(\llbracket x_i \rrbracket, \pi_i, \sigma_i)$  to every party.
2. **Round 2:** For each  $i \in [n]$ ,  $P_i$  sends all the messages it received in Round 1 to party  $P_n (= Q)$ .
  3. **Round 3:** Party  $P_n (= Q)$  does the following:
    - Define strings  $\text{msg}, \text{ct}_1, \dots, \text{ct}_n$  as  $\perp$ .
    - For each  $i \in [n]$ , let  $\{(\llbracket x_j \rrbracket^{i \rightarrow n}, \pi_j^{i \rightarrow n}, \sigma_j^{i \rightarrow n})\}_{j \in [n] \setminus \{i\}}$  denote the message received from  $P_i$  in Round 2 and  $(\llbracket x_i \rrbracket^{i \rightarrow n}, \pi_i^{i \rightarrow n}, \sigma_i^{i \rightarrow n})$  denote the message received from  $P_i$  in Round 1.
    - For each  $j \in [n]$ , do the following:
      - Let  $\{(\llbracket x_j \rrbracket^{1 \rightarrow n}, \pi_j^{1 \rightarrow n}, \sigma_j^{1 \rightarrow n}), \dots, (\llbracket x_j \rrbracket^{n \rightarrow n}, \pi_j^{n \rightarrow n}, \sigma_j^{n \rightarrow n})\}$  be the messages received across both rounds on behalf of party  $P_j$ .
      - Pick the lowest  $i_1$  such that  $\text{Verify}(\text{vkey}_j, (\llbracket x_j \rrbracket^{i_1 \rightarrow n}, \pi_j^{i_1 \rightarrow n}, \sigma_j^{i_1 \rightarrow n})) = 1$  and  $\text{NIZK.Verify}(\text{crs}, \pi_j^{i_1 \rightarrow n}, \text{st}_j) = 1$  for  $\text{st}_j \in \mathsf{L}_1$  where  $\text{st}_j = (\llbracket x_j \rrbracket^{i_1 \rightarrow n}, \text{pk})$ . Set  $\text{ct}_j := \llbracket x_j \rrbracket^{i_1 \rightarrow n}$  and  $\text{msg} := \text{msg} \parallel \text{"Party } j \text{"} \parallel (\llbracket x_j \rrbracket^{i_1 \rightarrow n}, \pi_j^{i_1 \rightarrow n}, \sigma_j^{i_1 \rightarrow n})$ .
      - If no such  $i_1$  exists, set  $\text{msg} = \text{msg} \parallel \text{"Party } j \text{"} \parallel \perp$ .
    - Compute  $\sigma_{\text{msg}} \leftarrow \text{Sign}(\text{skey}_n, \text{msg})$ . Send  $(\text{msg}, \sigma_{\text{msg}})$  to all parties.
    - Set  $\llbracket y \rrbracket = \text{dTFHE.Eval}(\text{pk}, f, \text{ct}_1, \dots, \text{ct}_n)$ .<sup>9</sup>
  4. **Round 4:** For each  $i \in [n-1]$ ,  $P_i$  sends the message received from  $Q$  in Round 3 to every party.
  5. **Round 5:** For each  $i \in [n-1]$ ,  $P_i$  does the following:
    - Let  $\{(\text{msg}^{j \rightarrow i}, \sigma_{\text{msg}}^{j \rightarrow i})\}_{j \in [n-1] \setminus \{i\}}$  be the messages received in Round 4 and  $(\text{msg}^{n \rightarrow i}, \sigma_{\text{msg}}^{n \rightarrow i})$  be the message from  $Q$  in Round 3.
    - If  $\text{Verify}(\text{vkey}_n, \text{msg}^{n \rightarrow i}, \sigma_{\text{msg}}^{n \rightarrow i}) \neq 1$  (OR)  $\text{msg}^{n \rightarrow i}$  is not of the form  $(\text{"Party 1 "} \parallel m_1 \parallel \dots \parallel \text{"Party } n \text{"} \parallel m_n)$ , send  $\perp$  to  $Q$  and end the round.
    - Output  $\perp$  to  $Q$  and end the round if there exists  $j \neq n$  such that:
      - $\text{msg}^{j \rightarrow i} \neq \text{msg}^{n \rightarrow i}$  (AND)
      - $\text{Verify}(\text{vkey}_n, \text{msg}^{j \rightarrow i}, \sigma_{\text{msg}}^{j \rightarrow i}) = 1$  (AND)
      - $\text{msg}^{j \rightarrow i}$  is of the form  $(\text{"Party 1 "} \parallel m_1, \dots, \parallel \text{"Party } n \text{"} \parallel m_n)$  This third check is to ensure that a corrupt  $P_j$  doesn't re-use a valid signature sent by  $Q$  in the first round as its message in Round 4.
    - Define strings  $\text{ct}_1, \dots, \text{ct}_n$ .
    - Parse  $\text{msg}^{n \rightarrow i}$  as  $(\text{"Party 1 "} \parallel m_1, \dots, \parallel \text{"Party } n \text{"} \parallel m_n)$ .
    - For each  $j \in [n]$ , do the following:
      - If in Round 1,  $P_i$  received  $(\llbracket x_j \rrbracket, \pi_j, \sigma_j)$  from  $P_j$  such that  $\text{Verify}(\text{vkey}_j, (\llbracket x_j \rrbracket, \pi_j), \sigma_j) = 1$  and  $\text{NIZK.Verify}(\pi_j, \text{st}_j) = 1$  for  $\text{st}_j \in \mathsf{L}_1$  where  $\text{st}_j = (\llbracket x_j \rrbracket, \text{pk})$ , set  $\text{bit}_j = 1$ . Else, set  $\text{bit}_j = 0$ .

<sup>9</sup> Let  $\mathcal{S} = \{i \mid \text{ct}_i = \perp\}$ . Here, we actually homomorphically evaluate the residual function  $f_{\mathcal{S}}(\cdot)$  that only takes as input  $\{x_j\}_{j \notin \mathcal{S}}$  and uses the default values for all indices in the set  $\mathcal{S}$ . For ease of exposition, we skip this notation in the rest of the protocol and proof.

- If  $m_j = \perp$ :
  - \* If  $\text{bit}_j = 1$ , send  $\perp$  to  $Q$  and end the round.
  - \* Else, set  $\text{ct}_j = \perp$ .
- If  $m_j = (\llbracket x_j \rrbracket^{i_1 \rightarrow n}, \pi_j^{i_1 \rightarrow n}, \sigma_j^{i_1 \rightarrow n})$ :
  - \* If  $\text{Verify}(\text{vkey}_j, (\llbracket x_j \rrbracket^{i_1 \rightarrow n}, \pi_j^{i_1 \rightarrow n}, \sigma_j^{i_1 \rightarrow n})) = 1$  and  $\text{NIZK.Verify}(\text{crs}, \pi_j^{i_1 \rightarrow n}, \text{st}_j) = 1$  for  $\text{st}_j \in L_1$  where  $\text{st}_j = (\llbracket x_j \rrbracket^{i_1 \rightarrow n}, \text{pk})$ , set  $\text{ct}_j = \llbracket x_j \rrbracket^{i_1 \rightarrow n}$ .
  - \* Else, send  $\perp$  to  $Q$  and end the round.
- Compute  $\llbracket y \rrbracket \leftarrow \text{dTFHE.Eval}(\text{pk}, f, \text{ct}_1, \dots, \text{ct}_n)$ .
- Compute  $\llbracket y : \text{sk}_i \rrbracket \leftarrow \text{dTFHE.PartialDec}(\text{sk}_i, \llbracket y \rrbracket)$  and  $\pi_i^{\text{dec}} \leftarrow \text{NIZK.Prove}(\text{crs}, \text{st}_i^{\text{dec}}, \text{wit}_i^{\text{dec}})$  for  $\text{st}_i^{\text{dec}} \in L_2$  where  $\text{st}_i^{\text{dec}} = (\llbracket y : \text{sk}_i \rrbracket, \llbracket y \rrbracket, \text{pk}_i, i)$  and  $\text{wit}_i^{\text{dec}} = (\text{sk}_i, r_i)$ .
- Send  $(\llbracket y : \text{sk}_i \rrbracket, \pi_i^{\text{dec}})$  to  $Q$ .

6. **Output Computation:**  $Q$  does the following:

- Recall the value  $\llbracket y \rrbracket$  computed in Round 3.
- For each  $i \in [n]$ , if  $\text{NIZK.Verify}(\text{crs}, \pi_i^{\text{dec}}, \text{st}_i^{\text{dec}}) \neq 1$  for  $\text{st}_i^{\text{dec}} \in L_2$  where  $\text{st}_i^{\text{dec}} = (\llbracket y : \text{sk}_i \rrbracket, \llbracket y \rrbracket, \text{pk}_i, i)$ , discard  $\llbracket y : \text{sk}_i \rrbracket$ .
- Output  $y \leftarrow \text{dTFHE.Combine}(\text{pk}, \{\llbracket y : \text{sk}_i \rrbracket\}_{i \in S})$  where  $S$  contains the set of non-discarded values from the previous step.

### 5.3 $(t + 2)$ Round Protocol

In this section, we elaborate on how a  $(t + 2)$ -round protocol for solitary MPC with  $\text{god}$ , say  $\Pi'$ , can be derived from the two-round protocol of [GLS15]. Recall that the two-round protocol (say  $\Pi$ ) of [GLS15] (that assumes a PKI setup) achieves  $\text{god}$  for standard MPC and involves communication only via broadcast channels in both rounds. We propose the following minor modifications to  $\Pi$ . First, we employ a  $(t + 1)$ -round protocol over pairwise-private channels that realizes the broadcast functionality [DS83] to execute Round 1 of  $\Pi$ . Next, the messages communicated via broadcast in Round 2 of  $\Pi$  are instead communicated privately only to  $Q$  (as only  $Q$  is supposed to obtain output) in Round  $(t + 2)$  of  $\Pi'$ . This completes the high-level description of  $\Pi'$  whose security follows directly from security of  $\Pi$ . Lastly, note that this approach achieves better round complexity than our general five-round construction from Section 5.2 only when  $t \leq 2$ .

## 6 Special Case: $t = 1$ with PKI and no Broadcast

In this section, we consider the special case of  $t = 1$  in the setting with a PKI setup and no broadcast. We assume that parties can communicate only via pairwise-private channels and have access to CRS and PKI. First, we present a lower bound that shows the necessity of three rounds to compute a 3-party solitary functionality involving an input from  $Q$ , assuming  $t = 1$ . This proves that the general  $(t + 2)$ -round construction of Section 5.3 is optimal when  $t = 1$  and  $Q$  has input. Next, we present a two-round upper bound for the case when  $Q$  does not have an input. This implies the tightness of the two-round lower bound of [HLP11] for the case when  $Q$  does not have input. Our lower bound holds even in the setting of a non-rushing adversary and our upper bounds hold even in the stronger adversarial setting of a rushing malicious adversary.

## 6.1 Necessity of Three Rounds When $Q$ Has Input

We show that in the absence of a broadcast channel, it is impossible to design a two-round 3-party solitary MPC protocol tolerating  $t = 1$  corruption that achieves **god**, even if parties are given access to CRS and PKI. Our lower bound holds even for non-rushing adversaries. Notably, it can also be extended to any  $n \geq 3$  and  $n/3 \leq t < n/2$ .

However, our lower bound argument crucially relies on the property that the function  $f$  to be computed involves an input provided by  $Q$ . Infact, this lower bound of three rounds can be circumvented when  $t = 1$  and  $f$  does not involve an input provided by  $Q$ , as demonstrated by our two-round solitary MPC with **god** in [Section 6.2.1](#) designed for the special case of  $t = 1$ .

**Theorem 6.1.** *Assume parties have access to CRS, PKI and pairwise-private channels. Let  $n$  and  $t$  be positive integers such that  $n \geq 3$  and  $n/3 \leq t < n/2$ . Then, there exists a solitary functionality  $f$  (involving input from  $Q$ ) such that no two-round  $n$ -party MPC protocol tolerating  $t$  corruptions can compute  $f$  with **god** even against a non-rushing adversary.*

*Proof.* We focus on the setting of  $n = 3$  and  $t = 1$  (which can be extended to  $n/3 \leq t < n/2$  using party partitioning, as elaborated in the proof of [Theorem 4.1](#)). Suppose for the sake of contradiction that there exists a two-round solitary MPC with **god**, say  $\Pi$  which computes a three-party solitary function  $f(x_1, x_2, x_3)$  among  $\{P_1, P_2, P_3\}$  where  $Q = P_3$  denotes the output receiving party providing an input  $x_3$ . Note that at most one of the three parties can be controlled by the adversary. Without loss of generality, we assume that Round 2 involves messages only from  $P_1$  and  $P_2$  to  $Q$  (as  $Q$  is the only party supposed to receive output at the end of Round 2).

Let  $f(x_1, x_2, x_3)$  where  $x_i \in \{0, 1\}$  for  $i \in [3]$  denotes  $P_i$ 's input be defined as

$$f(x_1, x_2, x_3) = \begin{cases} x_1 & \text{if } x_3 = 0 \\ x_2 & \text{if } x_3 = 1 \end{cases} .$$

The high-level structure of the proof is as follows - first, we claim that  $\Pi$  must be such that even if a corrupt party (either of  $P_1/P_2$ ) drops its Round 2 message (to  $Q$ ),  $Q$  must still be able to obtain the output. This lets us infer that  $\Pi$  in fact must be such that it allows a potentially corrupt  $Q$  to obtain two distinct evaluations of  $f$  based on two distinct inputs of its choice, which is the final contradiction. We now describe it formally.

We use the following notation: Let  $\text{pc}_{i \rightarrow j}^r$  denote the pairwise-private communication from  $P_i$  to  $P_j$  in round  $r$  where  $r \in [2], \{i, j\} \in [3]$ . These messages may be a function of the common reference string (denoted by  $\text{crs}$ ) and the PKI setup. Let  $\alpha_i$  denote the output of the PKI setup to party  $P_i$ . A party's view comprises of  $\text{crs}$ ,  $\alpha_i$ , its input, randomness and incoming messages.

**Lemma 6.2.**  *$\Pi$  must be such that an honest  $Q$  is able to compute the output with respect to its input (say  $x_3$ ) with overwhelming probability, even if one among  $P_1$  and  $P_2$  aborts in Round 2.*

*Proof.* The proof is straightforward - Suppose adversary corrupts one among  $P_1$  and  $P_2$ , say  $P_1$  who aborts in Round 2 (i.e. does not send  $\text{pc}_{1 \rightarrow 3}^2$ ). From the security of  $\Pi$  (guaranteed output delivery), it follows that an honest  $Q$  must still be able to obtain the correct output (even without  $\text{pc}_{1 \rightarrow 3}^2$ ) with respect to its input (say  $x_3$ ) with overwhelming probability.  $\square$

**Lemma 6.3.**  *$\Pi$  is such that it is possible for a potentially corrupt  $Q$  to obtain evaluations of  $f$  on  $x'_3$  as well as  $\widetilde{x}_3$  with overwhelming probability, for any choice of  $x'_3$  and  $\widetilde{x}_3$ .*

*Proof.* Consider a scenario where the adversary corrupts  $Q$  actively who does the following - in Round 1,  $Q$  behaves as per the protocol but using inputs  $x'_3$  and  $\widetilde{x}_3$  to send messages to  $P_2$  and  $P_1$  respectively. Note that this communication is over private channels in accordance with our network model. In Round 2,  $Q$  does not send any messages as per our assumption regarding the protocol design. We first claim that  $Q$  can obtain the output of  $f$  computed on  $x'_3$  as follows -  $Q$  discards the Round 2 message from  $P_1$  (i.e.  $\text{pc}_{1 \rightarrow 3}^2$ ). It is easy to see that this updated view of  $Q$  (after discarding the Round 2 private message from  $P_1$ ) is identically distributed to the scenario of [Lemma 6.2](#) where an honest  $Q$  used input  $x_3 = x'_3$  and  $P_1$  aborted in Round 2. It thus follows from [Lemma 6.2](#) that this view should enable  $Q$  to compute the output with respect to  $x'_3$  with overwhelming probability. Specifically, if  $x'_3 = 1$ , this would allow the adversary to learn  $x_2$  (i.e. the honest  $P_2$ 's input). Similarly, we can argue that by discarding the Round 2 private message from  $P_2$ ,  $Q$  is also able to learn the output with respect to  $\widetilde{x}_3$  with overwhelming probability. This is because his updated view (after discarding the Round 2 private message from  $P_1$ ) is identically distributed to the scenario of [Lemma 6.2](#) where an honest  $Q$  used input  $x_3 = \widetilde{x}_3$  and  $P_2$  aborted in Round 2. Specifically, if  $\widetilde{x}_3 = 0$ , this would allow the adversary to learn  $x_1$  (i.e. the honest  $P_1$ 's input). This completes the proof of the lemma that the adversary obtains multiple evaluations of  $f$ .  $\square$

We can thus conclude from [Lemma 6.3](#) that protocol  $\Pi$  is such that a corrupt  $Q$  can obtain two distinct evaluations of the function with respect to two choices of inputs  $x'_3$  and  $\widetilde{x}_3$  (where  $x'_3 \neq \widetilde{x}_3$ ), while the inputs of honest parties remain fixed. As elaborated above, if the adversary chooses  $x'_3 = 1$  and  $\widetilde{x}_3 = 0$ , then he learns both  $x_2$  and  $x_1$  with overwhelming probability which breaches security of  $\Pi$ . We have thus arrived at a contradiction to our assumption of  $\Pi$  being secure; thereby completing the proof of [Theorem 6.1](#).

**Circumvention of the lower bound.** First, we note that for scenarios where  $Q$  has input, the argument of multiple evaluations of  $f$ , holds only if at least one other party (different from  $Q$ ) also has input (which constitutes the non-trivial case, else  $Q$  could compute the output locally using just its input). Next, we point out that [Lemma 6.3](#) is meaningful only when  $Q$  has input, thereby our lower bound argument holds only in such a case. This is demonstrated by our 2-round upper bound in [Section 6.2.1](#) when  $Q$  does not have an input (for the special case of  $t = 1$ ).  $\square$

## 6.2 Protocols

For single corruption  $t = 1$ , we present a two-round protocol when  $Q$  does not have input in [Section 6.2.1](#) and a three-round protocol when  $Q$  has input in [Section 6.2.2](#).

### 6.2.1 Two-Round Protocol When $Q$ Has No Input

In this section, we present a two-round protocol for the setting where the receiving party  $Q$  does not have input and there is at most one corrupted party. Our protocol will utilize the following primitives: a 2-out-of- $n$  decentralized threshold FHE scheme  $\text{dTFHE} = (\text{dTFHE.DistGen}, \text{dTFHE.Enc}, \text{dTFHE.PartialDec}, \text{dTFHE.Eval}, \text{dTFHE.Combine})$ , a digital signature scheme  $(\text{Gen}, \text{Sign}, \text{Verify})$ , and a simulation-extractible NIZK argument  $(\text{NIZK.Setup}, \text{NIZK.Prove}, \text{NIZK.Verify})$ . We use the NIZK argument for two NP languages  $L_1, L_2$  defined in [Section 3.3.2](#). All of them can be built assuming  $\text{LWE}$  [[BGG<sup>+</sup>18](#), [CCH<sup>+</sup>19](#), [PS19](#)]. Formally, we show the following:

**Theorem 6.4.** *Assuming LWE, the two-round protocol described below achieves solitary output MPC with god with a PKI setup and pairwise-private channels. The protocol works for any  $n$ , any function where the receiving party  $Q$  does not have input and is secure against a malicious rushing adversary that can corrupt any one party.*

We consider  $n$  parties  $P_1, \dots, P_n$  who wish to evaluate function  $f : (\{0, 1\}^\lambda)^{n-1} \rightarrow \{0, 1\}^\lambda$ . We also denote  $P_n$  as the output receiving party  $Q$ . At a high level, our protocol works as follows. In Round 1, each party  $P_i$  sends to every other party a dTFHE encryption  $\llbracket x_i \rrbracket$  along with a NIZK argument proving that the encryption is well formed. On top of that,  $P_i$  also attaches its signature on the message. In Round 2, if party  $P_i$  detects dishonest behavior of another party in Round 1 (e.g., party  $P_j$  didn't communicate to  $P_i$ , or the message received from  $P_j$  does not contain a valid NIZK or signature), then it must be the case that  $Q$  is honest, so  $P_i$  sends  $x_i$  directly to  $Q$ . Here, we crucially rely on the fact that  $t = 1$ . Otherwise,  $P_i$  must have a valid set of ciphertexts  $\llbracket x_1 \rrbracket, \dots, \llbracket x_{n-1} \rrbracket$ .  $P_i$  can homomorphically compute the function  $f$  on the ciphertexts to obtain an encryption of the output  $\llbracket y \rrbracket$  and a partial decryption  $\llbracket y : \text{sk}_i \rrbracket$ .  $P_i$  sends all the ciphertexts and  $\llbracket y : \text{sk}_i \rrbracket$  (with NIZKs and signatures) to  $Q$ . Finally,  $Q$  either receives at least one set of valid ciphertexts along with a valid partial decryption, or receives at least  $n - 2$  inputs. In the first case,  $Q$  can compute a partial decryption of  $\llbracket y : \text{sk}_n \rrbracket$  by itself and combine the two partial decryptions to recover  $y$ . In the second case,  $Q$  can compute the output directly. Our protocol is formally described below. We defer the proof of security to [Appendix D](#).

**CRS:** Let  $\text{crs} \leftarrow \text{NIZK.Setup}(1^\lambda)$  be the common reference string.

**PKI Setup:**

- For each  $i \in [n]$ , generate  $(\text{pk}_i, \text{sk}_i) \leftarrow \text{dTFHE.DistGen}(1^\lambda, 1^d, i; r_i)$ ,  $(\text{vkey}_i, \text{skey}_i) \leftarrow \text{Gen}(1^\lambda)$ .
- Public keys:  $\text{pk} = (\text{pk}_1 \parallel \dots \parallel \text{pk}_n)$  and  $\{\text{vkey}_i\}_{i \in [n]}$ .
- Secret keys:  $(\text{sk}_i, r_i, \text{skey}_i)$  for party  $P_i$ .

**Inputs:** For each  $i \in [n - 1]$ , party  $P_i$  has an input  $x_i \in \{0, 1\}^\lambda$ .

**Protocol:**

- **Round 1:** For each  $i \in [n - 1]$ , party  $P_i$  does the following:
  1. Compute  $\llbracket x_i \rrbracket \leftarrow \text{dTFHE.Enc}(\text{pk}, x_i; \rho_i)$ .
  2. Compute  $\pi_i \leftarrow \text{NIZK.Prove}(\text{crs}, \text{st}_i, \text{wit}_i)$  for  $\text{st}_i \in \mathbf{L}_1$  where  $\text{st}_i = (\llbracket x_i \rrbracket, \text{pk})$  and  $\text{wit}_i = (x_i, \rho_i)$ .
  3. Compute  $\sigma_i \leftarrow \text{Sign}(\text{skey}_i, (\llbracket x_i \rrbracket, \pi_i))$ .
  4. Send  $(\llbracket x_i \rrbracket, \pi_i, \sigma_i)$  to every party  $P_j$  where  $j \in [n - 1] \setminus \{i\}$ .
- **Round 2:** For each  $i \in [n - 1]$ , party  $P_i$  does the following:
  1. For each  $j \in [n - 1] \setminus \{i\}$ , verify the following:
    - $P_i$  received  $(\llbracket x_j \rrbracket, \pi_j, \sigma_j)$  from party  $P_j$  in Round 1.
    - $\text{NIZK.Verify}(\text{crs}, \pi_j, \text{st}_j) = 1$ .
    - $\text{Verify}(\text{vkey}_j, (\llbracket x_j \rrbracket, \pi_j), \sigma_j) = 1$ .
  2. If any of the above isn't true, then send  $x_i$  to  $Q$ . Otherwise,
    - (a) Compute  $\llbracket y \rrbracket \leftarrow \text{dTFHE.Eval}(\text{pk}, f, \llbracket x_1 \rrbracket, \dots, \llbracket x_{n-1} \rrbracket)$ .

- (b) Compute  $\llbracket y : \text{sk}_i \rrbracket \leftarrow \text{dTFHE.PartialDec}(\text{sk}_i, \llbracket y \rrbracket)$ .
- (c) Compute  $\pi_i^{\text{dec}} \leftarrow \text{NIZK.Prove}(\text{crs}, \text{st}_i^{\text{dec}}, \text{wit}_i^{\text{dec}})$  for  $\text{st}_i^{\text{dec}} \in \mathsf{L}_2$  where  $\text{st}_i^{\text{dec}} = (\llbracket y : \text{sk}_i \rrbracket, \llbracket y \rrbracket, \text{pk}_i, i)$  and  $\text{wit}_i^{\text{dec}} = (\text{sk}_i, r_i)$ .
- (d) Send  $(\{\llbracket x_j \rrbracket, \pi_j, \sigma_j \rrbracket\}_{j \in [n-1]}, \llbracket y \rrbracket, \llbracket y : \text{sk}_i \rrbracket, \pi_i^{\text{dec}})$  to  $Q$ .

• **Output Computation:**  $Q$  does the following:

1. For each  $i \in [n-1]$ , verify the following:
  - $Q$  received  $(\{\llbracket x_j \rrbracket, \pi_j, \sigma_j \rrbracket\}_{j \in [n-1]}, \llbracket y \rrbracket, \llbracket y : \text{sk}_i \rrbracket, \pi_i^{\text{dec}})$  from party  $P_i$  in Round 2.
  - $\text{NIZK.Verify}(\text{crs}, \pi_j, \sigma_j) = 1$  and  $\text{Verify}(\text{vkey}_j, (\llbracket x_j \rrbracket, \pi_j), \sigma_j) = 1$  for all  $j \in [n-1]$ .
  - $\llbracket y \rrbracket = \text{dTFHE.Eval}(\text{pk}, f, \llbracket x_1 \rrbracket, \dots, \llbracket x_{n-1} \rrbracket)$ .
  - $\text{NIZK.Verify}(\text{crs}, \pi_i^{\text{dec}}, \text{st}_i^{\text{dec}}) = 1$ .
2. If the above is true for any  $i \in [n-1]$  (if it holds for multiple parties, pick the smallest  $i$ ), then
  - (a) Let  $(\{\llbracket x_j \rrbracket, \pi_j, \sigma_j \rrbracket\}_{j \in [n-1]}, \llbracket y \rrbracket, \llbracket y : \text{sk}_i \rrbracket, \pi_i^{\text{dec}})$  be the message  $Q$  received from  $P_i$  in Round 2.
  - (b) Compute  $\llbracket y : \text{sk}_n \rrbracket \leftarrow \text{dTFHE.PartialDec}(\text{sk}_n, \llbracket y \rrbracket)$ .
  - (c) Compute  $y \leftarrow \text{dTFHE.Combine}(\text{pk}, \{\llbracket y : \text{sk}_i \rrbracket, \llbracket y : \text{sk}_n \rrbracket\})$  and output  $y$ .
3. Otherwise,  $Q$  must have received  $x_i$  from  $P_i$  for at least  $n-2$  parties.
  - If  $Q$  received  $x_i$  from  $P_i$  for all  $i \in [n-1]$ , then output  $f(x_1, \dots, x_{n-1})$ .
  - Otherwise,  $Q$  did not receive the input from  $P_j$ , then output  $f(x_1, \dots, x_{j-1}, \hat{x}_j, x_{j+1}, \dots, x_{n-1})$ , where  $\hat{x}_j$  is the default input for  $P_j$ .

### 6.2.2 Three-Round Protocol When $Q$ Has Input

We note that for this special case of  $t = 1$  when  $Q$  has input, the general  $(t+2)$ -round construction of [Section 5.3](#) yields a three-round protocol with god.

## 7 Special Case: $t = 2$ with PKI and no Broadcast

In this section, we consider the special case of  $t = 2$  in the setting with a PKI setup and no broadcast. Once again, we assume that parties can communicate only via pairwise-private channels and have access to CRS and PKI. Our lower bound holds even in the setting of a non-rushing adversary and our upper bounds hold even in the stronger adversarial setting of a rushing malicious adversary.

### 7.1 Necessity of Three Rounds

In this section, we show that three rounds are necessary for 5-party solitary MPC with god against  $t = 2$  corruptions, even when  $Q$  has no input. This is in contrast to the case of  $t = 1$  for which two rounds are sufficient when  $Q$  has no input ([Section 6.2.1](#)). Interestingly, this lower bound of three rounds can be extended to any  $n \geq 5$  and  $2 \lceil n/5 \rceil \leq t < n/2$ . We state the formal theorem below.

**Theorem 7.1.** *Assume parties have access to CRS, PKI and pairwise-private channels. Let  $n$  and  $t$  be positive integers such that  $n \geq 5$  and  $2 \lceil n/5 \rceil \leq t < n/2$ . Then, there exists a solitary functionality  $f$  such that no two-round  $n$ -party MPC protocol tolerating  $t$  corruptions can compute  $f$  with god, even against a non-rushing adversary.*

*Proof.* We focus on the setting  $n = 5, t = 2$  (which can be extended to  $2 \lceil n/5 \rceil \leq t < n/2$  using party partitioning, as elaborated in the proof of [Theorem 4.7](#)). Suppose for the sake of contradiction that there exists a two-round five-party solitary MPC with god, say  $\Pi$  which is secure against  $t = 2$  corruptions. Let  $\Pi$  compute the solitary function  $f(x_1, x_2, x_3, x_4, x_5)$  among  $\{P_1, P_2, P_3, P_4, P_5\}$  where  $Q = P_5$  denotes the output receiving party. We clarify that our argument holds irrespective of whether  $f$  involves an input from  $Q$ . For simplicity, let  $f(x_1, x_2 = (x_2^0, x_2^1), x_3 = \perp, x_4 = (x_4^0, x_4^1), x_5 = \perp)$  be defined as

$$f(x_1, \dots, x_5) = \begin{cases} (x_2^0, x_4^0) & \text{if } x_1 = 0 \\ (x_2^1, x_4^1) & \text{if } x_1 = 1 \end{cases}$$

where  $x_1 \in \{0, 1\}$  and  $x_2, x_4 \in \{0, 1\}^2$ .

We consider an execution of  $\Pi$  with inputs  $(x_1, x_2, x_3, x_4, x_5)$  and analyze three different scenarios. Similar to [Section 6.1](#), we assume that Round 2 involves only messages sent to  $Q$  (as  $Q$  is the only party supposed to receive output at the end of Round 2).

In each of these scenarios, we assume that the adversary uses the honest input on behalf of the corrupt parties and its malicious behaviour is limited to dropping some of the messages supposed to be sent by the corrupt parties. Following is a description of the scenarios:

- Scenario 1:** Adversary corrupts  $\{P_2, P_3\}$  who behave honestly in Round 1 and simply abort (do not communicate) in Round 2.
- Scenario 2:** Adversary corrupts  $\{P_1, P_4\}$ .  $P_1$  does not communicate throughout the protocol.  $P_4$  behaves honestly in Round 1 and aborts in Round 2.
- Scenario 3:** Adversary corrupts  $\{P_1, Q\}$ .  $P_1$  communicates as per protocol steps only to  $P_4, Q$  in Round 1 (drops its private messages to  $P_2, P_3$ ). In Round 2,  $P_1$  behaves honestly i.e. communicates privately towards  $Q$  as per protocol steps.<sup>10</sup>  $Q$  behaves honestly throughout.

At a high-level, we first show that  $\Pi$  must be such that if Scenario 1 occurs, it must result in  $Q$  obtaining the correct output with respect to the input of honest  $P_1$  i.e.  $x_1$  with overwhelming probability. Next, we show that if Scenario 2 occurs, the output obtained by  $Q$  must be with respect to some  $x_1^*$ , that is independent of  $x_1$ , with non-negligible probability. Building on these properties of  $\Pi$ , we show that  $\Pi$  is such that an adversary corrupting  $\{P_1, Q\}$  can in fact get multiple evaluations of  $f$ , which is the final contradiction.

We present the views of the parties corresponding to each of these scenarios in [Tables 15-16](#). Let  $\text{pc}_{i \rightarrow j}^r$  denote the pairwise communication from  $P_i$  to  $P_j$  in Round  $r$  where  $r \in [2], \{i, j\} \in [5]$ . These messages may be a function of the common reference string (denoted by  $\text{crs}$ ) and the PKI setup. Let  $\alpha_i$  denote the output of the PKI setup to party  $P_i$ .  $\text{View}_i$  denotes the view of party  $P_i$  which comprises of  $\text{crs}$ ,  $\alpha_i$ , its input, randomness and incoming messages. The messages that  $P_2, P_3$  are supposed to send in Round 2 to  $Q$ , when they did not receive any communication from  $P_1$  in Round 1 (which could potentially be different from their communication in an all honest execution), are marked with  $\sim$  (such as  $\widetilde{\text{pc}}_{2 \rightarrow 5}^2$ ).

<sup>10</sup>In general, private communication between corrupt parties need not be specified. We mention this communication between corrupt parties only for clarity.

	Scenario 1					Scenario 2				
	View <sub>1</sub>	View <sub>2</sub>	View <sub>3</sub>	View <sub>4</sub>	View <sub>5</sub>	View <sub>1</sub>	View <sub>2</sub>	View <sub>3</sub>	View <sub>4</sub>	View <sub>5</sub>
Initial Input	$(x_1, r_1, \text{crs}, \alpha_1)$	$(x_2, r_2, \text{crs}, \alpha_2)$	$(x_3, r_3, \text{crs}, \alpha_3)$	$(x_4, r_4, \text{crs}, \alpha_4)$	$(x_5, r_5, \text{crs}, \alpha_5)$	$(x_1, r_1, \text{crs}, \alpha_1)$	$(x_1, r_2, \text{crs}, \alpha_2)$	$(x_3, r_3, \text{crs}, \alpha_3)$	$(x_4, r_4, \text{crs}, \alpha_4)$	$(x_5, r_5, \text{crs}, \alpha_5)$
Round 1	$\text{pc}_{2 \rightarrow 1}^1, \text{pc}_{3 \rightarrow 1}^1, \text{pc}_{4 \rightarrow 1}^1, \text{pc}_{5 \rightarrow 1}^1$	$\text{pc}_{1 \rightarrow 2}^1, \text{pc}_{3 \rightarrow 2}^1, \text{pc}_{4 \rightarrow 2}^1, \text{pc}_{5 \rightarrow 2}^1$	$\text{pc}_{1 \rightarrow 3}^1, \text{pc}_{2 \rightarrow 3}^1, \text{pc}_{4 \rightarrow 3}^1, \text{pc}_{5 \rightarrow 3}^1$	$\text{pc}_{1 \rightarrow 4}^1, \text{pc}_{2 \rightarrow 4}^1, \text{pc}_{3 \rightarrow 4}^1, \text{pc}_{5 \rightarrow 4}^1$	$\text{pc}_{1 \rightarrow 5}^1, \text{pc}_{2 \rightarrow 5}^1, \text{pc}_{3 \rightarrow 5}^1, \text{pc}_{4 \rightarrow 5}^1$	$\text{pc}_{2 \rightarrow 1}^1, \text{pc}_{3 \rightarrow 1}^1, \text{pc}_{4 \rightarrow 1}^1, \text{pc}_{5 \rightarrow 1}^1$	-, $\text{pc}_{3 \rightarrow 2}^1, \text{pc}_{4 \rightarrow 2}^1, \text{pc}_{5 \rightarrow 2}^1$	-, $\text{pc}_{2 \rightarrow 3}^1, \text{pc}_{4 \rightarrow 3}^1, \text{pc}_{5 \rightarrow 3}^1$	-, $\text{pc}_{2 \rightarrow 4}^1, \text{pc}_{3 \rightarrow 4}^1, \text{pc}_{5 \rightarrow 4}^1$	-, $\text{pc}_{2 \rightarrow 5}^1, \text{pc}_{3 \rightarrow 5}^1, \text{pc}_{4 \rightarrow 5}^1$
Round 2	-	-	-	-	$\text{pc}_{1 \rightarrow 5}^2, \text{pc}_{4 \rightarrow 5}^2$	-	-	-	-	$\text{pc}_{2 \rightarrow 5}^2, \text{pc}_{3 \rightarrow 5}^2$

Table 15: Views of  $P_1, P_2, P_3, P_4, Q$  in Scenario 1 & 2.

	Scenario 3				
	View <sub>1</sub>	View <sub>2</sub>	View <sub>3</sub>	View <sub>4</sub>	View <sub>5</sub>
Initial Input	$(x_1, r_1, \text{crs}, \alpha_1)$	$(x_2, r_2, \text{crs}, \alpha_2)$	$(x_3, r_3, \text{crs}, \alpha_3)$	$(x_4, r_4, \text{crs}, \alpha_4)$	$(x_5, r_5, \text{crs}, \alpha_5)$
Round 1	$\text{pc}_{2 \rightarrow 1}^1, \text{pc}_{3 \rightarrow 1}^1, \text{pc}_{4 \rightarrow 1}^1, \text{pc}_{5 \rightarrow 1}^1$	-, $\text{pc}_{3 \rightarrow 2}^1, \text{pc}_{4 \rightarrow 2}^1, \text{pc}_{5 \rightarrow 2}^1$	-, $\text{pc}_{2 \rightarrow 3}^1, \text{pc}_{4 \rightarrow 3}^1, \text{pc}_{5 \rightarrow 3}^1$	$\text{pc}_{1 \rightarrow 4}^1, \text{pc}_{2 \rightarrow 4}^1, \text{pc}_{3 \rightarrow 4}^1, \text{pc}_{5 \rightarrow 4}^1$	$\text{pc}_{1 \rightarrow 5}^1, \text{pc}_{2 \rightarrow 5}^1, \text{pc}_{3 \rightarrow 5}^1, \text{pc}_{4 \rightarrow 5}^1$
Round 2	-	-	-	-	$\text{pc}_{2 \rightarrow 5}^2, \text{pc}_{3 \rightarrow 5}^2, \text{pc}_{1 \rightarrow 5}^2, \text{pc}_{4 \rightarrow 5}^2$

Table 16: Views of  $P_1, P_2, P_3, P_4, Q$  in Scenario 3.

We now present a sequence of lemmas:

**Lemma 7.2.**  $\Pi$  is such that if Scenario 1 occurs, then  $Q$  obtains an output computed w.r.t.  $(x_1, x_4)$ , with all but negligible probability.

*Proof.* It follows from the security (in particular, correctness and god) of  $\Pi$  that  $Q$  obtains an output that is computed on the honest parties' inputs with all but negligible probability.  $\square$

**Lemma 7.3.**  $\Pi$  is such that if Scenario 2 occurs, then there exists  $x_1^* \in \{0, 1\}$  such that  $Q$  obtains an output computed w.r.t.  $x_1^*$  and  $x_2$  with non-negligible probability.

*Proof.* First, it follows from security of  $\Pi$  that  $Q$  must obtain an output computed on honest  $P_2$ 's input  $x_2$ , with all but negligible probability. Next, we note that Scenario 2 does not involve any communication from  $P_1$ , hence the output obtained by  $Q$  should be computed on some  $x_1'$  that is independent of  $x_1$ . Since there are only two possible values for  $x_1'$ , there must exist  $x_1^* \in \{0, 1\}$  such that the output is computed on  $x_1^*$  with non-negligible probability.  $\square$

Now we prove the final contradiction. Consider the special  $x_1^*$  in Lemma 7.3, which can be guessed by the adversary by probability at least 1/2. Suppose Scenario 3 occurs for  $(x_1, x_2, x_4)$  where  $x_1 = \neg x_1^*$ . First, we claim that  $Q$  can obtain the output of Scenario 1. Suppose  $Q$  discards

the Round 2 messages from  $P_2$  and  $P_3$  (i.e discards  $\widetilde{\text{pc}}_{2 \rightarrow 5}^2$  and  $\widetilde{\text{pc}}_{3 \rightarrow 5}^2$  from its view). Then, note that this updated view of  $Q$  is identically distributed to its view in Scenario 1 (refer Tables 15 - 16). Thus, from Lemma 7.2,  $Q$  learns the output based on  $\neg x_1^*$  and  $x_4$  with non-negligible probability. Next, we argue similarly that  $Q$  can obtain output of Scenario 2 as well. Suppose  $Q$  discards the Round 2 messages from  $P_1$  and  $P_4$  (i.e discards  $\text{pc}_{1 \rightarrow 5}^2$  and  $\text{pc}_{4 \rightarrow 5}^2$  from its view). Then, note that this updated view of  $Q$  is identically distributed to its view in Scenario 2 (refer Tables 15 - 16). From Lemma 7.3,  $Q$  learns output computed based on  $x_1^*$  and  $x_2$ .

To conclude,  $\Pi$  is such that an adversary corrupting  $\{P_1, Q\}$  can obtain an output computed on  $(\neg x_1^*, x_4)$  as well as an output computed on  $(x_1^*, x_2)$  with non-negligible probability, for any  $(x_2, x_4)$ . In particular, if  $x_1^* = 0$ , then the adversary can learn both  $x_4^1$  and  $x_2^0$  with non-negligible probability, which is not allowed as per the definition of  $f$ ; similarly, if  $x_1^* = 1$ , then the adversary can learn both  $x_4^0$  and  $x_2^1$  with non-negligible probability, which is also not allowed by  $f$ . This contradicts the security of  $\Pi$ ; completing the proof of Theorem 7.1.  $\square$

**Circumvention of the Lower Bound.** We point that the above lower bound can be circumvented if  $f$  involves only a single input. This is because multiple evaluations of  $f$  with respect to two different inputs of  $P_1$  (or any party other than  $Q$ ) leaks more information than  $Q$  is supposed to know only if the function involves at least two inputs (among parties excluding  $Q$ ). In fact, if this is not the case i.e. only one party among the set of parties excluding  $Q$  has an input (say  $P_i$ ), then a two-round protocol with `god` can be derived using a two-party non-interactive secure computation protocol (between  $P_i$  and  $Q$ ) as elaborated at the end of Section 4.1.

## 7.2 Four-Round Protocol

We note that for this special case of  $t = 2$ , the general  $(t + 2)$ -round construction of Section 5.3 yields a four-round protocol with `god`.

## References

- [ABMO15] Gilad Asharov, Amos Beimel, Nikolaos Makriyannis, and Eran Omri. Complete characterization of fairness in secure two-party computation of boolean functions. In *TCC*, 2015.
- [ACGJ18] Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Round-optimal secure multiparty computation with honest majority. CRYPTO, 2018.
- [ACJ17] Prabhanjan Ananth, Arka Rai Choudhuri, and Abhishek Jain. A new approach to round-optimal secure multiparty computation. CRYPTO, 2017.
- [ACOS20] Bar Alon, Ran Cohen, Eran Omri, and Tom Suad. On the power of an honest majority in three-party computation without broadcast. In *TCC*, 2020.
- [ADD<sup>+</sup>19] Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. Synchronous byzantine agreement with expected  $O(1)$  rounds, expected  $o(n^2)$  communication, and optimal resilience. In *Financial Cryptography and Data Security (FC)*, 2019.

- [AJL<sup>+</sup>12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. *EUROCRYPT*, 2012.
- [AMPR14] Arash Afshar, Payman Mohassel, Benny Pinkas, and Ben Riva. Non-interactive secure computation based on cut-and-choose. *EUROCRYPT*, 2014.
- [BBG<sup>+</sup>20] James Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrede Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly)logarithmic overhead. In *CCS*, pages 1253–1269. ACM, 2020.
- [BGG<sup>+</sup>18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. *CRYPTO*, 2018.
- [BGI<sup>+</sup>17] Saikrishna Badrinarayanan, Sanjam Garg, Yuval Ishai, Amit Sahai, and Akshay Wadia. Two-message witness indistinguishability and secure computation in the plain model from new assumptions. *ASIACRYPT*, 2017.
- [BGJ<sup>+</sup>17] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Dakshita Khurana, and Amit Sahai. Round optimal concurrent mpc via strong simulation. In *TCC*, 2017.
- [BGJ<sup>+</sup>18] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Amit Sahai. Promise zero knowledge and its applications to round optimal MPC. *CRYPTO*, 2018.
- [BHP17] Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. *TCC*, 2017.
- [BIK<sup>+</sup>17] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *CCS*, 2017.
- [BJMS20] Saikrishna Badrinarayanan, Aayush Jain, Nathan Manohar, and Amit Sahai. Threshold multi-key fhe and applications to round-optimal mpc. In *ASIACRYPT*, 2020.
- [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.
- [BLPV18] Fabrice Benhamouda, Huijia Lin, Antigoni Polychroniadou, and Muthuramakrishnan Venkatasubramanian. Two-round adaptively secure multiparty computation from standard assumptions. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 175–205. Springer, Heidelberg, November 2018.
- [BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *STOC*, 1990.

- [CCG<sup>+</sup>20] Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Round optimal secure multiparty computation from minimal assumptions. In *TCC*, 2020.
- [CCH<sup>+</sup>19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-shamir: from practice to theory. In *STOC*, 2019.
- [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.
- [CHOR16] Ran Cohen, Iftach Haitner, Eran Omri, and Lior Rotem. Characterization of secure multiparty computation without broadcast. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 596–616. Springer, Heidelberg, January 2016.
- [CJS14] Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical UC security with a global random oracle. *CCS*, 2014.
- [CL14] Ran Cohen and Yehuda Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. *ASIACRYPT*, 2014.
- [Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *STOC*, 1986.
- [CMS89] Benny Chor, Michael Merritt, and David B. Shmoys. Simple constant-time consensus protocols in realistic failure models. *J. ACM*, 1989.
- [COSV17] Michele Ciampi, Rafail Ostrovsky, Siniscalchi, and Ivan Visconti. Round-optimal secure two-party computation from trapdoor permutations. In *TCC*, 2017.
- [CPS20] T.-H. Hubert Chan, Rafael Pass, and Elaine Shi. Sublinear-round byzantine agreement under corrupt majority. *PKC*, 2020.
- [DMSY20] Ivan Damgård, Bernardo Magri, Luisa Siniscalchi, and Sophia Yakoubov. Broadcast-optimal two round MPC with an honest majority. *IACR Cryptol. ePrint Arch.*, 2020:1254, 2020.
- [DS83] Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 1983.
- [FG03] Matthias Fitzi and Juan A. Garay. Efficient player-optimal protocols for strong and differential consensus. In Elizabeth Borowsky and Sergio Rajsbaum, editors, *22nd ACM PODC*, pages 211–220. ACM, July 2003.
- [FGMO01] Matthias Fitzi, Juan A. Garay, Ueli M. Maurer, and Rafail Ostrovsky. Minimal complete primitives for secure multi-party computation, 2001.
- [FL82] Michael J. Fischer and Nancy A. Lynch. A lower bound for the time to assure interactive consistency. *Inf. Process. Lett.*, 1982.

- [FM89] Paul Feldman and Silvio Micali. An optimal probabilistic algorithm for synchronous byzantine agreement. ICALP, 1989.
- [FN09] Matthias Fitzi and Jesper Buus Nielsen. On the number of synchronous rounds sufficient for authenticated byzantine agreement. DISC, 2009.
- [GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 74–94. Springer, Heidelberg, February 2014.
- [GGJ19] Sanjam Garg, Aarushi Goel, and Abhishek Jain. The broadcast message complexity of secure multiparty computation. ASIACRYPT, 2019.
- [GHKL11] S. Dov Gordon, Carmit Hazay, Jonathan Katz, and Yehuda Lindell. Complete fairness in secure two-party computation. *J. ACM*, 58(6):24:1–24:37, 2011.
- [GIKR01] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. The round complexity of verifiable secret sharing and secure multicast. In *STOC*, 2001.
- [GIKR02] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. On 2-round secure multiparty computation. In *CRYPTO*, 2002.
- [GKKO07] Juan A. Garay, Jonathan Katz, Chiu-Yuen Koo, and Rafail Ostrovsky. Round complexity of authenticated broadcast with a dishonest majority. In *48th FOCS*, pages 658–668. IEEE Computer Society Press, October 2007.
- [GL05] Shafi Goldwasser and Yehuda Lindell. Secure multi-party computation without agreement. *J. Cryptology*, 2005.
- [GLS15] S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. In *CRYPTO*, 2015.
- [GMPP16] Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. EUROCRYPT, 2016.
- [GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, 1987.
- [Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In *STOC*, 2011.
- [GS17] Sanjam Garg and Akshayaram Srinivasan. Garbled protocols and two-round MPC from bilinear maps. In Chris Umans, editor, *58th FOCS*, pages 588–599. IEEE Computer Society Press, October 2017.
- [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.

- [HHPV18] Shai Halevi, Carmit Hazay, Antigoni Polychroniadou, and Muthuramakrishnan Venkatasubramanian. Round-optimal secure multi-party computation. *CRYPTO*, 2018.
- [HIK<sup>+</sup>19] Shai Halevi, Yuval Ishai, Eyal Kushilevitz, Nikolaos Makriyannis, and Tal Rabin. On fully secure MPC with solitary output. In *TCC*, 2019.
- [HLP11] Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. *CRYPTO*, 2011.
- [IKKP15] Yuval Ishai, Ranjit Kumaresan, Eyal Kushilevitz, and Anat Paskin-Cherniavsky. Secure computation with minimal interaction, revisited. *CRYPTO*, 2015.
- [IKO<sup>+</sup>11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. *EUROCRYPT*, 2011.
- [IKP10] Yuval Ishai, Eyal Kushilevitz, and Anat Paskin. Secure multiparty computation with minimal interaction. *CRYPTO*, 2010.
- [KK09] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. *J. Comput. Syst. Sci.*, 2009.
- [KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *CRYPTO*, 2004.
- [KOS03] Jonathan Katz, Rafail Ostrovsky, and Adam D. Smith. Round efficiency of multi-party computation with a dishonest majority. In *EUROCRYPT*, 2003.
- [KY86] Anna Karlin and Andrew Yao. Probabilistic lower bounds for byzantine agreement. *Unpublished document*, 1986.
- [LSP82] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 1982.
- [Lyn96] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.
- [MR17] Payman Mohassel and Mike Rosulek. Non-interactive secure 2pc in the offline/online and batch settings. *EUROCRYPT*, 2017.
- [MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. *EUROCRYPT*, 2016.
- [MZ13] Ueli Maurer and Vassilis Zikas. Information-theoretic secure multiparty computation. *Secure Multi-Party Computation, Cryptology and Information Security Series*, 2013.
- [MZ17] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *IEEE S & P*, 2017.
- [PR18] Arpita Patra and Divya Ravi. On the exact round complexity of secure three-party computation. *CRYPTO*, 2018.
- [PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In *CRYPTO*, 2019.

- [PSL80] Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 1980.
- [Wee10] Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *FOCS*, 2010.
- [WXSD20] Jun Wan, Hanshen Xiao, Elaine Shi, and Srinivas Devadas. Expected constant round byzantine broadcast under dishonest majority. In *TCC*, 2020.
- [Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). *FOCS*, 1986.

## A Literature Survey

In this section, we discuss some additional related work relevant to our problem. We categorize them according to the security guarantee (**god**, fairness, with abort etc.), setup assumption (plain, CRS or PKI), type of communication channels used (pairwise-private or broadcast), number of allowed corruption (honest majority or not).

Round complexity of MPC has been explored copiously in the past in different settings as demonstrated by a plethora of works, e.g. [BMR90, GIKR02, KOS03, KO04, Wee10, Goy11, AJL<sup>+</sup>12, GGHR14, GLS15, GMPP16, MW16, BHP17, ACJ17, COSV17, BGJ<sup>+</sup>17, GS17, HHPV18, BGJ<sup>+</sup>18, GS18, BLPV18, BL18, ACGJ18, BJMS20, CCG<sup>+</sup>20, CGZ20]). This long line of works culminated recently with round optimal (two in CRS or PKI, four in the plain model) protocols from minimal assumptions [GS18, BL18, CCG<sup>+</sup>20].

Gennaro et al. [GIKR02] showed a necessity of three rounds for fair MPC in the CRS model, when the number of corruptions is at least 2 (i.e.  $t \geq 2$ ). This result holds irrespective of the total number of parties ( $n$ ) and assumes parties have access to a broadcast channel (and pairwise-private channels). Recently, Patra and Ravi [PR18] complement this (necessity of three rounds) for any  $t \geq n/3$  (including  $t = 1$ ). Gordon et al. [GLS15] showed a necessity of three rounds for fairness in CRS model assuming only broadcast channels (no pairwise-private channels). In the honest majority setting, the same work proposed a three round (optimal) MPC protocol achieving **god** in the CRS model. Assuming PKI setup, their protocol can be collapsed to two rounds matching the lower bound of [HLP11]. Recently, a couple of works [BJMS20, ACGJ18] improved this state-of-art by constructing three round protocols in the *plain model*. It is easy to see that a standard MPC protocol implies a protocol for solitary MPC (Section 4.4). Therefore, these results directly provide an upper bound of three rounds in our setting. In this paper we prove that this is in fact tight (Theorem 4.1). For a small number of parties, round optimal MPC protocols with **god** appear in [IKP10, IKKP15, PR18].

The above feasibility results on round-optimal MPC with **god** assume broadcast channels as the default mode of communication. However, broadcast channels are expensive [FL82, DS83] to implement in practice. This motivated works like [CHOR16] to characterize MPC without broadcast (or PKI). A recent work by Cohen et al. [CGZ20] explored the (im)possibility of two round (optimal) standard MPC protocols with minimal use of broadcast in the dishonest majority setting guaranteeing security with (different types of) abort, a weaker guarantee than **god**. Our setting is different from theirs as we focus on solitary MPC with **god** and therefore require honest majority [HIK<sup>+</sup>19]. The work of Damgård et al. [DMSY20] studies broadcast optimal two-round standard MPC in honest majority setting, assuming correlated randomness (such as PKI). This is in contrast to our work that assumes either broadcast or PKI (but not both) in addition to CRS.

The study of solitary MPC was recently initiated by Halevi et al. [HIK<sup>+</sup>19]. Standard MPC with **god** in the presence of a dishonest majority was already shown to be impossible [Cle86]. Their work focuses on constructing solitary MPC with **god** without honest majority. They show that computing arbitrary solitary functionalities is impossible too and provide positive results for various interesting classes of solitary functionalities such as private set intersection (PSI). Inspired by their work, we look into the complementary direction of building generic solitary MPC in the honest majority setting with a focus on round complexity. Another recent work [ACOS20] studied fully secure solitary MPC without broadcast (or PKI), in that they necessarily (due to the aforementioned impossibility) focus on restricted classes of functionality.

Even assuming PKI, achieving standard MPC (with **god**) requires  $(t + 1)$  rounds [DS83] deter-

ministically. We note that a separate line of research [GKKO07, FN09, CPS20, WXSD20] bypasses the  $(t + 1)$  bound for broadcast by requiring standard MPC protocols to run in *expected* constant rounds – however, it is not true that the protocol terminates in constant rounds with non-negligible probability. In contrast, our focus is on designing protocols that successfully terminate in constant rounds (with probability one).

## B Necessity of Broadcast/PKI

In this section, we sketch a proof that solitary MPC with **god** cannot be achieved by pairwise-private channels alone even in the CRS model assuming honest majority. In particular, when  $n/3 \leq t < n/2$  (where  $n$  is the total number of parties and  $t$  is the number of corrupted parties), there exist certain functions with solitary output that cannot be securely computed with **god**. Therefore, a broadcast channel or PKI setup is necessary. A similar argument was presented in [FGMO01] for the functionality of converge-cast, which only works for *information theoretic* security. A recent work of Alon et al. [ACOS20] studies which functions can be computed by a 3-party MPC protocol with **god** in the *plain model* with no broadcast channel and no PKI setup, and shows a specific solitary functionality that cannot be computed. Both arguments also work in our setting even in the presence of a CRS. For sake of completeness we present the proof below.

We first consider the special case of  $n = 3$  and  $t = 1$ , and then reduce the general case of  $n \geq 3$  and  $t \geq n/3$  to this special case.

**Lemma B.1.** *Let  $n = 3$ . There exist functions with solitary output that cannot be securely computed with **god** given pairwise-private channels even in the CRS model when  $t = 1$ .*

*Proof.* Suppose, for the sake of contradiction, that there is a protocol  $\Pi$  that can securely compute any function with **god** for three parties  $P_1, P_2, Q$  where  $Q$  is the party receiving the output, even if one of the parties is maliciously corrupted.

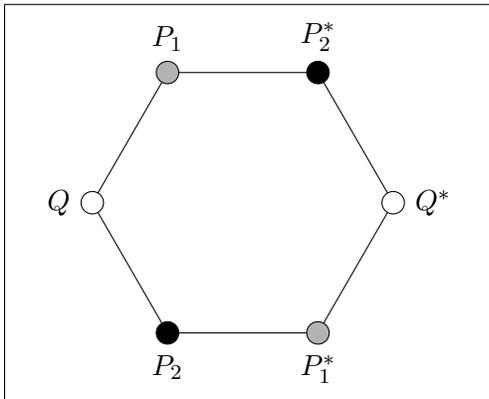


Figure 1: Rearrangement of parties.

Let  $P_1^*, P_2^*, Q^*$  be identical copies of  $P_1, P_2, Q$ , respectively. We build a network involving all six parties by arranging them in a circle as shown in Figure 1. Each party communicates with their adjacent parties following the protocol  $\Pi$ .

We claim that for every pair of adjacent parties in the circle, their common view can be thought of as the view of two honest parties in  $\Pi$  with respect to a malicious adversary that corrupts the remaining party. For example, the view of  $(P_1, P_2^*)$  in the new system is the same as the view of honest  $(P_1, P_2)$  against a malicious  $Q$  where  $Q$ 's strategy is to “split” itself into  $Q$  and  $Q^*$  and then simulate the communication between  $P_1$  and  $Q$  as well as between  $P_2^*$  and  $Q^*$ .

For an arbitrary function  $f$ , let  $P_1, P_2, Q, P_1^*, P_2^*, Q^*$  hold inputs  $x_1, x_2, x_3, x_1^*, x_2^*, x_3^*$ , respectively. If we consider the pair of parties  $(P_1, Q)$  in the circle, then their view is the same as the two honest parties against a malicious  $P_2$ . By the **god** property of the protocol  $\Pi$ ,  $Q$  should output  $f(x_1, x_2', x_3)$  computed on  $P_1$  and  $Q$ 's honest inputs and some  $x_2'$ . On the other hand, if we consider the pair of parties  $(P_2^*, Q^*)$  in the circle, then their view is

the same as the two honest parties against a malicious  $P_1$ . By the `god` property of  $\Pi$ ,  $Q^*$  should output  $f(x'_1, x_2^*, x_3^*)$  computed on honest inputs of  $P_2^*$  and  $Q^*$  and on some  $x'_1$ .

Now consider the pair of parties  $(P_1, P_2^*)$  in the circle, their view is the same as the two honest parties  $(P_1, P_2^*)$  with inputs  $(x_1, x_2^*)$  against a malicious  $Q$ . If the adversary splits itself into  $Q$  and  $Q^*$  and then simulates the communication in the circle, then it learns both  $f(x_1, x_2', x_3)$  and  $f(x'_1, x_2^*, x_3^*)$ .

Consider a function  $f(x_1, x_2, x_3)$  defined as follows:

$$f(x_1, x_2, x_3) = \begin{cases} x_1 & \text{if } x_3 = 0 \\ x_2 & \text{if } x_3 = 1 \end{cases}$$

where  $x_1, x_2, x_3 \in \{0, 1\}$ . A malicious party  $Q$  can learn both  $x_1$  and  $x_2^*$  by setting  $x_3 = 0$  and  $x_3^* = 1$ , which violates the security guarantees resulting in a contradiction.  $\square$

**Theorem B.2.** *Let  $n \geq 3$ . There exist functions with solitary output that cannot be securely computed with `god` given pairwise-private channels even in the CRS model if  $t \geq n/3$ .*

*Proof.* Suppose, for the sake of contradiction, that there is a protocol  $\Pi$  that can securely compute any function with `god` for  $n$  parties where  $t \geq n/3$  parties are maliciously corrupted.

Then we can let three parties  $P_1, P_2, Q$  each simulate up to  $\lceil n/3 \rceil$  of the parties in  $\Pi$ , with the receiving party simulated by  $Q$ . Thus, the new protocol among parties  $P_1, P_2, Q$  achieves secure MPC with `god` against one corrupted party because it simulates at most  $\lceil n/3 \rceil$  parties in  $\Pi$ , which is tolerated by assumption. Since this contradicts [Lemma B.1](#), the theorem follows.  $\square$

## C Security Proof For Five-Round Protocol

In this section, we formally prove [Theorem 5.7](#). Let  $\text{NIZK.Sim} = (\text{NIZK.Sim.Setup}, \text{NIZK.Sim.Prove}, \text{NIZK.Sim.Ext})$  denote the straight-line simulator for the simulation-extractible NIZK argument. Consider a malicious adversary  $\mathcal{A}$  that corrupts a set of  $t$  parties where  $t < n/2$ . Let  $\mathcal{H}$  denote the set of honest parties and  $\mathcal{S}^*$  the set of corrupt parties.

**Simulation Strategy.** We now describe the strategy of the simulator `Sim`.

**CRS:** Compute  $(\text{simcrs}, \text{td}) \leftarrow \text{NIZK.Sim.Setup}(1^\lambda)$ . Send `simcrs` to  $\mathcal{A}$  for every corrupt party  $P_i$ .

**PKI Setup:**

- For each  $i \in [n]$ , sample  $(\text{pk}_i, \text{sk}_i) \leftarrow \text{dTFHE.DistGen}(1^\lambda, 1^d, i; r_i)$  and  $(\text{vkey}_i, \text{skey}_i) \leftarrow \text{Gen}(1^\lambda)$ .
- Public key:  $\text{pk} = \text{pk}_1 \parallel \dots \parallel \text{pk}_n$  and  $\{\text{vkey}_i\}_{i \in [n]}$ .
- Secret keys:  $(\text{sk}_i, r_i, \text{skey}_i)$  for party  $P_i$  for each  $i \in [n]$ .
- Send the public key and corresponding secret keys to  $\mathcal{A}$  for every corrupt party  $P_i$ .

We consider two cases of the corrupted parties. In the first case  $Q$  is honest and in the second case  $Q$  is corrupted.

**Case 1: Honest  $Q$ .** We now describe the simulator's strategy if the output receiving party  $P_n = Q$  is honest.

1. **Round 1:** For each honest party  $P_i \in \mathcal{H}$ :

- Compute  $\llbracket 0_i \rrbracket \leftarrow \text{dTFHE.Enc}(\text{pk}, 0^\lambda)$  using fresh randomness,  $\pi_i \leftarrow \text{NIZK.Sim.Prove}(\text{td}, \text{st}_i)$  for  $\text{st}_i \in \mathcal{L}_1$  where  $\text{st}_i = (\llbracket 0_i \rrbracket, \text{pk})$ .
  - Compute  $\sigma_i \leftarrow \text{Sign}(\text{skey}_i, (\llbracket 0_i \rrbracket, \pi_i))$ . Send  $(\llbracket 0_i \rrbracket, \pi_i, \sigma_i)$  to  $\mathcal{A}$  for every corrupt party.
  - Receive a message  $(\llbracket x_j \rrbracket^{j \rightarrow i}, \pi_j^{j \rightarrow i}, \sigma_j^{j \rightarrow i})$  from  $\mathcal{A}$  for every corrupt party  $P_j$ .
2. **Round 2:** From  $\mathcal{A}$ , for every corrupt  $P_j$ , receive  $\{(\llbracket x_k \rrbracket^{j \rightarrow n}, \pi_k^{j \rightarrow n}, \sigma_k^{j \rightarrow n})\}_{k \in [n] \setminus \{j\}}$ . Also, for ease of notation, let's assume that each honest party  $P_i$  sends the messages it received from  $\mathcal{A}$  in Round 1 to  $Q$ . Let's denote this by  $\{(\llbracket x_j \rrbracket^{i \rightarrow n}, \pi_j^{i \rightarrow n}, \sigma_j^{i \rightarrow n})\}_{P_j \in \mathcal{S}^*}$
3. **Round 3:** For party  $P_n (= Q)$ , do following:
- Define strings  $\text{msg}, \text{ct}_1, \dots, \text{ct}_n$  as  $\perp$ .
  - Also, define strings  $\{\text{inp}_j\}_{P_j \in \mathcal{S}^*}$ .
  - For each corrupt party  $P_j$ , do the following:
    - Let  $\{(\llbracket x_j \rrbracket^{1 \rightarrow n}, \pi_j^{1 \rightarrow n}, \sigma_j^{2 \rightarrow n}), \dots, (\llbracket x_j \rrbracket^{n \rightarrow n}, \pi_j^{n \rightarrow n}, \sigma_j^{n \rightarrow n})\}$  be the message received across both rounds on behalf of party  $P_j$ . This includes messages sent by  $\mathcal{A}$  in Round 1 from  $P_j$  to every honest party (that was assumed to be forwarded to  $Q$  in Round 2 for ease of notation) and the messages sent by  $\mathcal{A}$  from other corrupt parties to  $Q$  in Round 2.
    - Pick the smallest index  $i_1$  such that  $\text{Verify}(\text{vkey}_j, (\llbracket x_j \rrbracket^{i_1 \rightarrow n}, \pi_j^{i_1 \rightarrow n}, \sigma_j^{i_1 \rightarrow n})) = 1$  and  $\text{NIZK.Verify}(\text{simcrs}, \pi_j^{i_1 \rightarrow n}, \text{st}_j) = 1$  for  $\text{st}_j \in \mathcal{L}_1$  where  $\text{st}_j = (\llbracket x_j \rrbracket^{i_1 \rightarrow n}, \text{pk})$ . Then,:
      - \* **Input Extraction and ZK Abort:** Let  $(\text{inp}_j, \rho_j) \leftarrow \text{NIZK.Sim.Ext}(\text{td}, \pi_j^{i_1 \rightarrow n}, \text{st}_j)$ . Output “ZK Abort” if  $\text{NIZK.Sim.Ext}(\cdot)$  outputs  $\perp$ .
      - \* Set  $\text{msg} := \text{msg} \parallel \text{“Party } j \text{”} \parallel (\llbracket x_j \rrbracket^{i_1 \rightarrow n}, \pi_j^{i_1 \rightarrow n}, \sigma_j^{i_1 \rightarrow n})$ . Also, set  $\text{ct}_j := \llbracket x_j \rrbracket^{i_1 \rightarrow n}$ .
    - If no such  $i_1$  exists, set  $\text{msg} := \text{msg} \parallel \text{“Party } j \text{”} \parallel \perp$  and  $\text{inp}_j = \hat{x}_j^{11}$ .
  - For each honest party  $P_i \in \mathcal{H}$ , do the following:
    - Set  $\text{msg} := \text{msg} \parallel \text{“Party } i \text{”} \parallel (\llbracket 0_i \rrbracket, \pi_i, \sigma_i)$  where  $(\llbracket 0_i \rrbracket, \pi_i, \sigma_i)$  is the tuple output in Round 1.
    - Also, set  $\text{ct}_i = \llbracket 0_i \rrbracket$ .
  - Compute  $\sigma_{\text{msg}} \leftarrow \text{Sign}(\text{skey}_n, \text{msg})$ . Send  $(\text{msg}, \sigma_{\text{msg}})$  to  $\mathcal{A}$  for every corrupt party.
  - Set  $\llbracket y \rrbracket = \text{dTFHE.Eval}(\text{pk}, f, \text{ct}_1, \dots, \text{ct}_n)$ .
4. **Round 4:** For each honest party  $P_i \in \mathcal{H}$ , for every corrupt party  $P_j \in \mathcal{S}^*$ :
- Send the same message as in Round 3 to  $\mathcal{A}$ .
  - Receive  $(\text{msg}^{j \rightarrow i}, \sigma_{\text{msg}}^{j \rightarrow i})$  from  $\mathcal{A}$ .
  - **Signature Abort:** Output “Signature Abort” if :
    - $\text{msg}^{j \rightarrow i} \neq \text{msg}$  (AND)
    - $\text{msg}^{j \rightarrow i}$  is of the form  $(\text{“Party } 1 \text{”} \parallel m_1 \parallel \dots \parallel \text{“Party } n \text{”} \parallel m_n)$  (AND)
    - $\text{Verify}(\text{vkey}_n, \text{msg}^{j \rightarrow i}, \sigma_{\text{msg}}^{j \rightarrow i}) = 1$
5. **Round 5:** From  $\mathcal{A}$ , for every corrupt party  $P_j$ , receive  $(\llbracket y : \text{sk}_j \rrbracket, \pi_j^{\text{dec}})$ .

---

<sup>11</sup>This denotes the default input for  $P_j$

6. **Query to Ideal Functionality  $\mathcal{F}$ :**

- **ZK Abort:** Output “ZK Abort” if  $\text{NIZK.Verify}(\text{simcrs}, \pi_j^{\text{dec}}, \text{st}_j^{\text{dec}}) = 1$  (AND)  
 $\text{NIZK.Sim.Ext}(\text{td}, \pi_j^{\text{dec}}, \text{st}_j^{\text{dec}}) = \perp$  for any  $j$  where  $\text{st}_j^{\text{dec}} = (\llbracket y : \text{sk}_j \rrbracket, \llbracket y \rrbracket, \text{pk}_j, j)$ .
- Send  $\{\text{inp}_j\}_{P_j \in \mathcal{S}^*}$  to  $\mathcal{F}$ .

**Case 2: Corrupt  $Q$ .** We now describe the simulator’s strategy if the output receiving party  $P_n = Q$  is corrupt.

1. **Round 1:** Same as Round 1 when  $Q$  is honest. That is, for each honest party  $P_i \in \mathcal{H}$ , for every corrupt party  $P_j$ , send  $(\llbracket 0_i \rrbracket, \pi_i, \sigma_i)$  to  $\mathcal{A}$  and receive  $(\llbracket x_j \rrbracket^{j \rightarrow i}, \pi_j^{j \rightarrow i}, \sigma_j^{j \rightarrow i})$ .
2. **Round 2:** For each honest party  $P_i \in \mathcal{H}$ , send the following to  $\mathcal{A}$  for corrupt party  $Q$ :
  - The set of messages received from  $\mathcal{A}$  to  $P_i$  in Round 1:  $\{(\llbracket x_j \rrbracket^{j \rightarrow i}, \pi_j^{j \rightarrow i}, \sigma_j^{j \rightarrow i})\}_{P_j \in \mathcal{S}^*}$ .
  - The set of messages  $\{(\llbracket 0_k \rrbracket, \pi_k, \sigma_k)\}_{P_k \in \mathcal{H}}$  generated in Round 1.
3. **Round 3:** For each honest party  $P_i \in \mathcal{H}$ , receive  $(\text{msg}^{n \rightarrow i}, \sigma_{\text{msg}}^{n \rightarrow i})$  from  $\mathcal{A}$  for party  $Q$ .
4. **Round 4:** For each honest party  $P_i \in \mathcal{H}$ , for every corrupt party  $P_j$ , send the message received in Round 3 -  $(\text{msg}^{n \rightarrow i}, \sigma_{\text{msg}}^{n \rightarrow i})$  to  $\mathcal{A}$  and receive  $(\text{msg}^{j \rightarrow i}, \sigma_{\text{msg}}^{j \rightarrow i})$ .
5. **Round 5:**

- Define set  $\mathcal{H}_1 = \mathcal{H}$  to be the set of parties that would send valid partial decryptions.
- **Pruning down  $\mathcal{H}_1$ :** For each  $P_i \in \mathcal{H}_1$ , do the following:
  - Let  $\{(\text{msg}^{n \rightarrow k}, \sigma_{\text{msg}}^{n \rightarrow k})\}_{k \in \mathcal{H}}$  be the Round 3 messages from  $\mathcal{A}$  and  $\{(\text{msg}^{j \rightarrow i}, \sigma_{\text{msg}}^{j \rightarrow i})\}_{j \in \mathcal{S}^*}$  be the message from  $\mathcal{A}$  to  $P_i$  in Round 4.
  - If  $\text{Verify}(\text{vkey}_n, \text{msg}^{n \rightarrow i}, \sigma_{\text{msg}}^{n \rightarrow i}) \neq 1$  (OR)  $\text{msg}^{n \rightarrow i}$  is not of the form (“Party 1 ”  $\| m_1 \| \dots \|$  “Party n ”  $\| m_n$ ), send  $\perp$  to  $\mathcal{A}$  from  $P_i$  (for party  $Q$ ) and remove  $P_i$  from  $\mathcal{H}_1$ .
  - Send  $\perp$  to  $\mathcal{A}$  from  $P_i$  (for party  $Q$ ) and remove  $P_i$  from  $\mathcal{H}_1$  if there exists  $k \neq i \in \mathcal{H}$  such that:
    - \*  $\text{msg}^{n \rightarrow k} \neq \text{msg}^{n \rightarrow i}$  (AND)
    - \*  $\text{Verify}(\text{vkey}_n, \text{msg}^{n \rightarrow k}, \sigma_{\text{msg}}^{n \rightarrow k}) = 1$  (AND)
    - \*  $\text{msg}^{n \rightarrow k}$  is of the form (“Party 1 ”  $\| m_1 \| \dots \|$  “Party n ”  $\| m_n$ ).
  - Send  $\perp$  to  $\mathcal{A}$  from  $P_i$  (for party  $Q$ ) and remove  $P_i$  from  $\mathcal{H}_1$  if there exists  $j \neq n \in \mathcal{S}^*$  such that:
    - \*  $\text{msg}^{j \rightarrow i} \neq \text{msg}^{n \rightarrow i}$  (AND)
    - \*  $\text{Verify}(\text{vkey}_n, \text{msg}^{j \rightarrow i}, \sigma_{\text{msg}}^{j \rightarrow i}) = 1$  (AND)
    - \*  $\text{msg}^{j \rightarrow i}$  is of the form (“Party 1 ”  $\| m_1 \| \dots \|$  “Party n ”  $\| m_n$ ).
- Define strings  $\text{ct}_1, \dots, \text{ct}_n, \{\text{inp}_j\}_{P_j \in \mathcal{S}^*}$  to be  $\perp$ .
- Let  $(\text{msg}, \sigma_{\text{msg}})$  be the unique Round 3 message received by all honest parties from  $\mathcal{A}$  where  $\text{msg}$  is of the form (“Party 1 ”  $\| m_1 \| \dots \|$  “Party n ”  $\| m_n$ ).
- **Query to Ideal Functionality  $\mathcal{F}$ .** If  $\mathcal{H}_1 \neq \emptyset$ , do:
  - For each honest party  $P_i \in \mathcal{H}$ , let  $m_i = (a_i, b_i, c_i)$ .

- \* **Signature Abort:** Output “Signature Abort” if  $(a_i, b_i) \neq (\llbracket 0_i \rrbracket, \pi_i)$  (AND)  $\text{Verify}(\text{vkey}_i, (a_i, b_i), c_i) = 1$ .
- \* If  $(a_i, b_i) \neq (\llbracket 0_i \rrbracket, \pi_i)$  or  $\text{Verify}(\text{vkey}_i, (a_i, b_i), c_i) \neq 1$ , send  $\perp$  to  $\mathcal{A}$  from all parties in  $\mathcal{H}_1$  and end the round.
- \* Else, set  $\text{ct}_i = \llbracket 0_i \rrbracket$ .
- For each corrupt party  $P_j \in \mathcal{S}^*$  with  $m_j = \perp$ :
  - \* **Pruning  $\mathcal{H}_1$  - Part Two:** If there exists  $P_i \in \mathcal{H}$  such that  $\text{Verify}(\text{vkey}_j, (\llbracket x_j \rrbracket^{j \rightarrow i}, \pi_j^{j \rightarrow i}), \sigma_j^{j \rightarrow i}) = 1$  and  $\text{NIZK.Verify}(\pi_j^{j \rightarrow i}, \text{st}_j) = 1$  for  $\text{st}_j \in \mathbf{L}_1$  where  $\text{st}_j = (\llbracket x_j \rrbracket^{j \rightarrow i}, \text{pk})$ , send  $\perp$  to  $\mathcal{A}$  from party  $P_i$  and remove  $P_i$  from  $\mathcal{H}_1$ . Here, recall that  $(\llbracket x_j \rrbracket^{j \rightarrow i}, \pi_j^{j \rightarrow i}, \sigma_j^{j \rightarrow i})$  is the message from  $\mathcal{A}$  to  $P_i$  in Round 1.
  - \* Else, set  $(\text{ct}_j, \text{inp}_j) = (\perp, \hat{x}_j)$ .
- For each corrupt party  $P_j \in \mathcal{S}^*$  with  $m_j = (\llbracket x_j \rrbracket, \pi_j, \sigma_j)$ :
  - \* If  $\text{Verify}(\text{vkey}_j, (\llbracket x_j \rrbracket, \pi_j), \sigma_j) \neq 1$  (or)  $\text{NIZK.Verify}(\text{crs}, \pi_j, \text{st}_j) \neq 1$  for  $\text{st}_j \in \mathbf{L}_1$  where  $\text{st}_j = (\llbracket x_j \rrbracket, \text{pk})$ , send  $\perp$  to  $\mathcal{A}$  from all parties in  $\mathcal{H}_1$  and end the round.
  - \* **Input Extraction and ZK Abort:** Let  $(\text{inp}_j, r_j) \leftarrow \text{NIZK.Sim.Ext}(\text{td}, \pi_j, \text{st}_j)$ . Output “ZK Abort” if  $\text{NIZK.Sim.Ext}(\cdot)$  outputs  $\perp$ .
  - \* Set  $\text{ct}_j = \llbracket x_j \rrbracket$ .
- Send  $\{\text{inp}_j\}_{P_j \in \mathcal{S}^*}$  to  $\mathcal{F}$  and receive output  $y$ .
- Compute  $\{\llbracket y : \text{sk}_i \rrbracket\}_{P_i \in \mathcal{H}} \leftarrow \text{dTFHE.Sim}(f, y, \text{ct}_1, \dots, \text{ct}_n, \{\text{sk}_j\}_{j \in \mathcal{S}^*})$
- For each honest party  $P_i \in \mathcal{H}_1$  (the ones that did not output  $\perp$ ):
  - Compute  $\pi_i^{\text{dec}} \leftarrow \text{NIZK.Sim.Prove}(\text{td}, \text{st}_i^{\text{dec}})$  for  $\text{st}_i^{\text{dec}} \in \mathbf{L}_2$  where  $\text{st}_i^{\text{dec}} = (\llbracket y : \text{sk}_i \rrbracket, \llbracket y \rrbracket, \text{pk}_i, i)$ ,  $\llbracket y \rrbracket = \text{dTFHE.Eval}(\text{pk}, f, \text{ct}_1, \dots, \text{ct}_n)$ .
  - Send  $(\llbracket y : \text{sk}_i \rrbracket, \pi_i^{\text{dec}})$  to  $Q$ .

**Hybrids.** We now show that the above simulation strategy is successful via a series of computationally indistinguishable hybrids where the first hybrid  $\text{hyb}_0$  corresponds to the real world and the last hybrid  $\text{hyb}_{10}$  corresponds to the ideal world.

- **hyb<sub>0</sub> - Real World.** In this hybrid, consider a simulator  $\text{Sim.hyb}$  that plays the role of the honest parties as in the real world.
- **hyb<sub>1</sub> - Simulate ZK.** In this hybrid,  $\text{Sim.hyb}$  first generates a simulated CRS in the setup phase:  $(\text{simcrs}, \text{td}) \leftarrow \text{NIZK.Sim.Setup}(1^\lambda)$ . Then, in Round 1 of both cases, it computes simulated ZK proofs:  $\pi_i \leftarrow \text{NIZK.Sim.Prove}(\text{td}, \text{st}_i)$ . Finally,  $\text{Sim.hyb}$  also computes simulated ZK proofs in Round 5 when  $Q$  is corrupt:  $\pi_i^{\text{dec}} \leftarrow \text{NIZK.Sim.Prove}(\text{td}, \text{st}_i^{\text{dec}})$ .
- **hyb<sub>2</sub> - Case 1: Signature Abort.** In this hybrid, when  $Q$  is honest, in Round 4, on behalf of every honest party  $P_i$ ,  $\text{Sim.hyb}$  runs the “Signature Abort” step as done by  $\text{Sim}$ . That is, output “Signature Abort” if the adversary is able to forge a valid signature on behalf of honest party  $Q$ .
- **hyb<sub>3</sub> - Case 1: Input Extraction and ZK Abort.** In this hybrid, when  $Q$  is honest, in Round 3,  $\text{Sim.hyb}$  runs the input extraction and ZK Abort step as done by  $\text{Sim}$  in the ideal world.  $\text{Sim.hyb}$  also runs the “ZK Abort” step in Round 5 as done by  $\text{Sim}$ . That is, in both steps, output “ZK Abort” if  $\text{NIZK.Sim.Ext}(\cdot)$  outputs  $\perp$ .

- **hyb<sub>4</sub> - Case 1: Query to ideal functionality.** In this hybrid, Sim.hyb sends the values  $\{\text{inp}_j\}_{P_j \in \mathcal{S}^*}$  extracted by running  $\text{NIZK.Sim.Ext}(\cdot)$  in Round 3 to  $\mathcal{F}$ . Further,  $\mathcal{F}$  delivers output to honest  $Q$ .
- **hyb<sub>5</sub> - Case 2: Pruning down  $\mathcal{H}_1$ .** In this hybrid, when  $Q$  is corrupt, in Round 5, Sim.hyb runs the pruning down  $\mathcal{H}_1$  step as done by Sim in the ideal world. That is, send  $\perp$  in Round 5 on behalf of corresponding honest parties that detect inconsistent signatures from  $Q$  across rounds 3 and 4.
- **hyb<sub>6</sub> - Case 2: Signature Abort.** In this hybrid, when  $Q$  is corrupt, in Round 5, Sim.hyb runs the “Signature Abort” step as done by Sim. That is, output “Signature Abort” if the adversary is able to forge a valid signature on behalf of any honest party.
- **hyb<sub>7</sub> - Case 2: Pruning  $\mathcal{H}_1$  - Part Two.** In this hybrid, when  $Q$  is corrupt, in Round 5, Sim.hyb runs the second part of pruning down  $\mathcal{H}_1$  step as done by Sim in the ideal world. That is, send  $\perp$  in Round 5 on behalf of honest parties that detect a missing ciphertext in the message sent by  $Q$  for which they had received a valid ciphertext in round one.
- **hyb<sub>8</sub> - Case 2: Input Extraction, ZK Abort and Query to ideal functionality.** In this hybrid, when  $Q$  is corrupt, in Round 5, Sim.hyb computes  $(\text{inp}_j, \text{ct}_j)$  as done by Sim in the ideal world. To do so, Sim.hyb also runs the “ZK Abort” step - that is, output “ZK Abort” if  $\text{NIZK.Sim.Ext}(\cdot)$  outputs  $\perp$ . Finally, queries  $\mathcal{F}$  with  $\{\text{inp}_j\}_{P_j \in \mathcal{S}^*}$  and receive output  $y$ .
- **hyb<sub>9</sub> - Case 2: Simulate Partial Decryptions.** In this hybrid, when  $Q$  is corrupt, in Round 5, Sim.hyb simulates the partial decryptions generated by the honest parties as done in the ideal world. That is, compute  $\{\llbracket y : \text{sk}_i \rrbracket\}_{P_i \in \mathcal{H}} \leftarrow \text{dTFHE.Sim}(f, y, \text{ct}_1, \dots, \text{ct}_n, \{\text{sk}_j\}_{j \in \mathcal{S}^*})$ .
- **hyb<sub>10</sub> - Switch Encryptions.** Finally, in this hybrid, in both cases, in Round 1, on behalf of every honest party  $P_i$ , Sim.hyb now computes  $\llbracket 0_i \rrbracket \leftarrow \text{dTFHE.Enc}(\text{pk}, 0^\lambda)$  instead of encrypting the honest party’s input. This corresponds to the ideal world.

We now show that every pair of consecutive hybrids is computationally indistinguishable.

**Lemma C.1.** *Assuming the zero knowledge property of the NIZK,  $\text{hyb}_0$  is computationally indistinguishable from  $\text{hyb}_1$ .*

*Proof.* The only difference between the two hybrids is that in  $\text{hyb}_0$ , the simulator Sim.hyb generates the NIZK proofs in rounds one and five on behalf of the honest parties (and the CRS) as in the real world by running the honest prover algorithm while in  $\text{hyb}_1$ , the proofs are simulated using the NIZK simulator  $\text{NIZK.Sim.Prove}(\cdot)$  (and the simulated CRS  $\text{simcrs}$  is generated using the simulator  $\text{NIZK.Sim.Setup}(\cdot)$ ). It is easy to observe that if there exists an adversary  $\mathcal{A}$  that can distinguish between these two hybrids with some non-negligible probability  $\epsilon$ , we can come up with a reduction  $\mathcal{A}_{\text{NIZK}}$  that can break the zero knowledge property of the NIZK argument which is a contradiction.  $\square$

**Lemma C.2.** *Assuming the security of the signature scheme,  $\text{hyb}_1$  is computationally indistinguishable from  $\text{hyb}_2$ .*

*Proof.* The only difference between the two hybrids is if Sim.hyb outputs “Signature Abort” in Round 4 of  $\text{hyb}_2$  with non-negligible probability which doesn’t happen in  $\text{hyb}_1$ . This can happen if, with non-negligible probability, in Round 4, on behalf of some corrupt party,  $\mathcal{A}$  sends a valid

message-signature pair of the right form that was not the one received from honest  $Q$  in Round 3. In more detail, this happens if, with non-negligible probability,  $\mathcal{A}$  sends a tuple  $(\text{msg}^{j \rightarrow i}, \sigma_{\text{msg}}^{j \rightarrow i})$  from corrupt party  $P_j$  to honest party  $P_i$  in Round 4 such that:

- $\text{msg}^{j \rightarrow i} \neq \text{msg}$  (AND)
- $\text{msg}^{j \rightarrow i}$  of the form (“Party 1 ”  $\parallel m_1 \parallel \dots \parallel$  “Party n ”  $\parallel m_n$ ) (AND)
- $\text{Verify}(\text{vkey}_n, (\text{msg}^{j \rightarrow i}, \sigma_{\text{msg}}^{j \rightarrow i})) = 1$

However, if this happens, we can build a reduction  $\mathcal{A}_{\text{Sign}}$  that breaks the unforgeability of the signature scheme which is a contradiction.  $\square$

**Lemma C.3.** *Assuming the simulation-extractibility property of the NIZK,  $\text{hyb}_2$  is computationally indistinguishable from  $\text{hyb}_3$ .*

*Proof.* The only difference between the two hybrids is if  $\text{Sim.hyb}$  outputs “ZK Abort” in Round 3 or 5 of  $\text{hyb}_3$  with non-negligible probability. This happens if, on behalf of some corrupt party  $P_j$ :

- In Round 1,  $\mathcal{A}$  sends  $(\llbracket x_j \rrbracket, \pi_j, \sigma_j)$  such that the signature verifies,  $\text{NIZK.Verify}(\text{simcrs}, \pi_j, \text{st}_j) = 1$  but  $\text{NIZK.Sim.Ext}(\text{td}, \pi_j, \text{st}_j) = \perp$  where  $\text{st}_j = (\llbracket x_j \rrbracket, \text{pk})$  (OR)
- In Round 5,  $\mathcal{A}$  sends  $(\llbracket y : \text{sk}_j \rrbracket, \pi_j^{\text{dec}})$  such that  $\text{NIZK.Verify}(\text{simcrs}, \pi_j^{\text{dec}}, \text{st}_j^{\text{dec}}) = 1$  but  $\text{NIZK.Sim.Ext}(\text{td}, \pi_j^{\text{dec}}, \text{st}_j^{\text{dec}}) = \perp$  where  $\text{st}_j = (\llbracket y : \text{sk}_j \rrbracket, \llbracket y \rrbracket, \text{pk}_j, j)$ .

However, from the security of the UC-secure NIZK, we have:

$$\Pr[\pi \leftarrow \mathcal{A}(\text{simcrs}, \text{st}); (\text{simcrs}, \text{td}) \leftarrow \text{NIZK.Sim.Setup}(1^\lambda) \text{ s.t.}$$

$$\text{NIZK.Verify}(\text{crs}, \pi, x) = 1 \text{ (AND) } \text{NIZK.Sim.Ext}(\text{td}, \pi, \text{st}) \perp] \leq \text{negl}(\lambda)$$

Thus, if there exists such an adversary that can cause this bad event to occur with non-negligible probability, we can build a reduction  $\mathcal{A}_{\text{NIZK}}$  that breaks the simulation-extractibility property of the NIZK argument which is a contradiction.  $\square$

**Lemma C.4.** *Assuming the correctness of the threshold FHE scheme and the NIZK,  $\text{hyb}_3$  is computationally indistinguishable from  $\text{hyb}_4$ .*

*Proof.* The only difference between the two hybrids is the manner in which honest  $Q$  learns the output. In  $\text{hyb}_4$ ,  $Q$  learns output  $y = f(\text{inp}_1, \dots, \text{inp}_n)$  from the ideal functionality  $\mathcal{F}$  where  $\text{inp}_i = x_i$  for every honest party  $P_i \in \mathcal{H}$  and  $\text{inp}_j = x_j$  (or  $\hat{x}_j$  for every corrupt party  $P_j \in \mathcal{S}^*$ ). From the correctness of the extractor  $\text{NIZK.Sim.Ext}$ , we know that if  $\text{NIZK.Sim.Ext}(\cdot)$  output  $\text{inp}_j$ , then indeed  $\text{inp}_j = x_j$ . Also, if  $\text{inp}_j$  is set to  $\hat{x}_j$ , then this is because  $\text{Sim.hyb}$  detected no valid tuple  $(\llbracket 0_j \rrbracket, \pi_j, \sigma_j)$  on behalf of party  $P_j$ .

In  $\text{hyb}_3$ ,  $Q$  computes the output as in the real world by following the protocol. Observe that due to the presence of an honest majority, from the correctness of the threshold FHE scheme,  $Q$  does recover output  $y' = f(\text{inp}'_1, \dots, \text{inp}'_n)$ . For every honest party  $P_i \in \mathcal{H}$ , it is easy to observe that  $\text{inp}'_i = x_i$ . For every corrupt party  $P_j \in \mathcal{S}^*$ , as in  $\text{hyb}_4$ , from the protocol description in Round 3, observe that  $\text{inp}_j = \hat{x}_j$ , if  $Q$  detects no valid tuple  $(\llbracket 0_j \rrbracket, \pi_j, \sigma_j)$  on behalf of party  $P_j$ . Hence the two hybrids are computationally indistinguishable.  $\square$

**Lemma C.5.**  $\text{hyb}_4$  is identically distributed to  $\text{hyb}_5$ .

*Proof.* Observe that in Round 5 of the simulation, the list of steps performed in the “pruning down  $\mathcal{H}_1$ ” part are in fact identical to the steps performed by all honest parties in the real world to check that they received one single valid  $(\text{msg}, \sigma_{\text{msg}})$  from  $Q$  in Round 3. Thus, the two hybrids are identical.  $\square$

**Lemma C.6.** Assuming the security of the signature scheme,  $\text{hyb}_5$  is computationally indistinguishable from  $\text{hyb}_6$ .

*Proof.* The only difference between the two hybrids is if  $\text{Sim.hyb}$  outputs “Signature Abort” in Round 5 of  $\text{hyb}_6$  with non-negligible probability. This can happen if, with non-negligible probability, in Round 3, on behalf of corrupt party  $Q$ , the tuple that  $\mathcal{A}$  sends includes a valid message-signature pair for some honest party  $P_i$  that was not the pair sent by  $P_i$  in Round 1. In more detail, for every honest party  $P_i \in \mathcal{H}$ , let  $(\llbracket 0_i \rrbracket, \pi_i, \sigma_i)$  be the message sent in Round 1. The bad event happens if, with non-negligible probability, in Round 3,  $\mathcal{A}$  sends  $(\text{msg}, \sigma_{\text{msg}})$  to every honest party where  $\text{Verify}(\text{vkey}_n, \text{msg}, \sigma_{\text{msg}}) = 1$ ,  $\text{msg}$  of the form (“Party 1 ”  $\|m_1\| \dots \|$  “Party n ”  $\|m_n$ ) and there exists honest party  $P_i \in \mathcal{H}$  such that:  $m_i = (a_i, b_i, c_i)$  (AND)  $(a_i, b_i) \neq (\llbracket 0_i \rrbracket, \pi_i)$  (AND)  $\text{Verify}(\text{vkey}_i, (a_i, b_i), c_i) = 1$ .

However, if this happens, we can build a reduction  $\mathcal{A}_{\text{Sign}}$  that breaks the unforgeability of the signature scheme which is a contradiction.  $\square$

**Lemma C.7.**  $\text{hyb}_6$  is identically distributed to  $\text{hyb}_7$ .

*Proof.* Observe that in Round 5 of the simulation, the list of steps performed in the “second part of pruning down  $\mathcal{H}_1$ ” are in fact identical to the step performed by all honest parties  $P_i$  in the real world to check that  $Q$  did not fail to include a ciphertext on behalf of some party  $P_k$  if the honest party did receive a valid ciphertext from  $P_k$  in Round one. Thus, the two hybrids are identical.  $\square$

**Lemma C.8.** Assuming the simulation-extractibility property of the NIZK,  $\text{hyb}_7$  is computationally indistinguishable from  $\text{hyb}_8$ .

*Proof.* This is identical to the proof of [Lemma C.3](#).  $\square$

**Lemma C.9.** Assuming the simulation security of the threshold FHE scheme,  $\text{hyb}_8$  is computationally indistinguishable from  $\text{hyb}_9$ .

*Proof.* The only difference between the two hybrids is that in  $\text{hyb}_8$ , the simulator  $\text{Sim.hyb}$  generates the partial decryptions of the threshold FHE scheme on behalf of the honest parties as in the real world while in  $\text{hyb}_9$ , they are simulated by running the simulator  $\text{dTFHE.Sim}$ . It is easy to observe that if there exists an adversary  $\mathcal{A}$  that can distinguish between these two hybrids with some non-negligible probability  $\epsilon$ , we can come up with a reduction  $\mathcal{A}_{\text{dTFHE}}$  that can break the simulation security of the threshold FHE scheme which is a contradiction.  $\square$

**Lemma C.10.** Assuming the semantic security of the threshold FHE scheme,  $\text{hyb}_9$  is computationally indistinguishable from  $\text{hyb}_{10}$ .

*Proof.* The only difference between the two hybrids is that in  $\text{hyb}_9$ , the simulator  $\text{Sim.hyb}$  generates the encryptions of the threshold FHE scheme on behalf of the honest parties as in the real world while in  $\text{hyb}_{10}$ , they are generated as encryptions of 0. It is easy to observe that if there exists an adversary  $\mathcal{A}$  that can distinguish between these two hybrids with some non-negligible probability  $\epsilon$ , we can come up with a reduction  $\mathcal{A}_{\text{dTFHE}}$  that can break the semantic security of the threshold FHE scheme which is a contradiction.  $\square$

## D Security Proof for $t = 1$ When $Q$ Has No Input

We now prove [Theorem 6.4](#). Let  $\text{NIZK.Sim} = (\text{NIZK.Sim.Setup}, \text{NIZK.Sim.Prove}, \text{NIZK.Sim.Ext})$  denote the straight-line simulator for the simulation-extractible NIZK argument. Consider a malicious adversary  $\mathcal{A}$  that corrupts a single party  $P_c$ . We construct an ideal-world PPT simulator  $\text{Sim}$  that interacts with  $\mathcal{A}$  as follows.

In the setup phase,  $\text{Sim}$  generates  $(\text{simcrs}, \text{td}) \leftarrow \text{NIZK.Sim.Setup}(1^\lambda)$  and follows the PKI setup honestly to generate the public and secret keys.  $\text{Sim}$  first invokes  $\mathcal{A}$  with the simulated CRS  $\text{simcrs}$ , public keys  $(\text{pk}, \{\text{vkey}_i\}_{i \in [n]})$ , and secret keys  $(\text{sk}_c, r_c, \text{skey}_c)$ . We consider two cases of the corrupted party. In the first case  $Q$  is honest and in the second case  $Q$  is corrupted.

**Case 1:  $Q$  is honest.** The corrupted party is  $P_c$  where  $c \in [n-1]$ . The strategy of  $\text{Sim}$  is described as follows:

- Let  $\tilde{x} := \perp$  and  $k := \infty$ .
- **Round 1:** For each  $i \in [n-1] \setminus \{c\}$ , do the following:
  - Compute the following:
    - $\llbracket 0_i \rrbracket \leftarrow \text{dTFHE.Enc}(\text{pk}, 0_i)$ .
    - $\pi_i \leftarrow \text{NIZK.Sim.Prove}(\text{td}, \text{st}_i)$  for  $\text{st}_i \in \mathbf{L}_1$  where  $\text{st}_i = (\llbracket 0_i \rrbracket, \text{pk})$ .
    - $\sigma_i \leftarrow \text{Sign}(\text{skey}_i, (\llbracket 0_i \rrbracket, \pi_i))$ .
  - Send  $(\llbracket 0_i \rrbracket, \pi_i, \sigma_i)$  to  $\mathcal{A}$  on behalf of party  $P_i$ .
  - Receive  $(\llbracket x_c \rrbracket^{c \rightarrow i}, \pi_c^{c \rightarrow i}, \sigma_c^{c \rightarrow i})$  from  $\mathcal{A}$  on behalf of party  $P_i$ .
  - If  $i < k$ ,  $\text{NIZK.Verify}(\text{simcrs}, \pi_c^{c \rightarrow i}, \text{st}_c^{c \rightarrow i}) = 1$  for  $\text{st}_c^{c \rightarrow i} \in \mathbf{L}_1$  where  $\text{st}_c^{c \rightarrow i} = (\llbracket x_c \rrbracket^{c \rightarrow i}, \text{pk})$ , and  $\text{Verify}(\text{vkey}_c, (\llbracket x_c \rrbracket^{c \rightarrow i}, \pi_c^{c \rightarrow i}), \sigma_c^{c \rightarrow i}) = 1$ , then
    - \* Decrypt  $\llbracket x_c \rrbracket^{c \rightarrow i}$  by running algorithms  $\text{dTFHE.PartialDec}(\cdot)$ ,  $\text{dTFHE.Combine}(\cdot)$  using dTFHE secret keys to obtain  $x_c^{c \rightarrow i}$ .
    - \* Let  $\tilde{x} := x_c^{c \rightarrow i}$  and  $k := i$ .
- **Round 2:**

If  $\text{Sim}$  receives  $x_c^{c \rightarrow n}$  from  $\mathcal{A}$  on behalf of  $Q$ , then let  $\tilde{x} := x_c^{c \rightarrow n}$  if  $k = \infty$ .

If  $\text{Sim}$  receives  $(\{(\llbracket x_j \rrbracket^{c \rightarrow n}, \pi_j^{c \rightarrow n}, \sigma_j^{c \rightarrow n})\}_{j \in [n-1]}, \llbracket y \rrbracket, \llbracket y : \text{sk}_c \rrbracket, \pi_c^{\text{dec}})$  from  $\mathcal{A}$  on behalf of  $Q$ , then verify the following:

- $\text{NIZK.Verify}(\text{simcrs}, \pi_j^{c \rightarrow n}, \text{st}_j^{c \rightarrow n}) = 1$  for  $\text{st}_j^{c \rightarrow n} \in \mathbf{L}_1$  where  $\text{st}_j^{c \rightarrow n} = (\llbracket x_j \rrbracket^{c \rightarrow n}, \text{pk})$  for all  $j \in [n-1]$ .
- $\text{Verify}(\text{vkey}_j, (\llbracket x_j \rrbracket^{c \rightarrow n}, \pi_j^{c \rightarrow n}), \sigma_j^{c \rightarrow n}) = 1$  for all  $j \in [n-1]$ .
- $\llbracket y \rrbracket = \text{dTFHE.Eval}(\text{pk}, f, \llbracket x_1 \rrbracket^{c \rightarrow n}, \dots, \llbracket x_{n-1} \rrbracket^{c \rightarrow n})$ .
- $\text{NIZK.Verify}(\text{simcrs}, \pi_c^{\text{dec}}, \text{st}_c^{\text{dec}}) = 1$ .

If all the above is true, then

- Decrypt  $\llbracket x_c \rrbracket^{c \rightarrow n}$  by dTFHE secret keys to obtain  $x_c^{c \rightarrow n}$ .
- Let  $\tilde{x} := x_c^{c \rightarrow n}$  if  $c < k$ .

- Finally, send  $\tilde{x}$  to the ideal functionality.

**Hybrid argument.** Now we show that  $\mathcal{A}$  and  $Q$ 's output in the real world is computationally indistinguishable from Sim and  $Q$ 's output in the ideal world.

- $\text{hyb}_0$ :  $\mathcal{A}$  and  $Q$ 's output in the real world.
- $\text{hyb}_1$ : First generate  $(\text{simcrs}, \text{td}) \leftarrow \text{NIZK.Sim.Setup}(1^\lambda)$  and give  $\text{simcrs}$  to  $\mathcal{A}$  as the CRS. In Round 1, for each  $i \in [n-1] \setminus \{c\}$ , replace  $\pi_i$  sent to  $\mathcal{A}$  by a simulated NIZK argument  $\text{NIZK.Sim.Prove}(\text{td}, \text{st}_i)$ .

By the zero-knowledge property of NIZK,  $\mathcal{A}$  cannot distinguish between  $\text{hyb}_0$  and  $\text{hyb}_1$ .

- $\text{hyb}_2$ : If  $Q$  receives a valid tuple  $(\{(\llbracket x_j \rrbracket, \pi_j, \sigma_j)\}_{j \in [n-1]}, \llbracket y \rrbracket, \llbracket y : \text{sk}_i \rrbracket, \pi_i^{\text{dec}})$  from party  $P_i$  in Round 2 (if it holds for multiple parties, then pick the smallest  $i$ ), then decrypt  $\{\llbracket x_j \rrbracket\}_{j \in [n-1]}$  by dTFHE secret keys to obtain  $(\tilde{x}_1, \dots, \tilde{x}_{n-1})$ . If the decryption fails, then abort.

We argue that the probability of abort is negligible. First we know that  $\{\llbracket x_j \rrbracket\}_{j \in [n-1] \setminus \{c\}}$  are all well-formed encryptions. By the simulation-extractability of NIZK for  $\mathbf{L}_1$ , we can run  $(\hat{x}_c, \hat{\rho}_c) \leftarrow \text{NIZK.Sim.Ext}(\text{td}, \pi_c, \text{st}_c)$  to extract  $(\hat{x}_c, \hat{\rho}_c)$  with all but negligible probability, hence  $\llbracket x_c \rrbracket$  is also a well-formed encryption. By the correctness of dTFHE, all the ciphertexts  $\{\llbracket x_j \rrbracket\}_{j \in [n-1]}$  can be decrypted with all but negligible probability.

- $\text{hyb}_3$ : Same as  $\text{hyb}_2$  except that after obtaining  $(\tilde{x}_1, \dots, \tilde{x}_{n-1})$  from dTFHE decryption,  $Q$  outputs  $f(\tilde{x}_1, \dots, \tilde{x}_{n-1})$ .

By the simulation-extractability of NIZK for  $\mathbf{L}_2$ , we can extract  $(\hat{\text{sk}}_c, \hat{r}_c) \leftarrow \text{NIZK.Sim.Ext}(\text{td}, \pi_c^{\text{dec}}, \text{st}_c^{\text{dec}})$  with all but negligible probability, hence  $\llbracket y : \text{sk}_c \rrbracket$  is correctly computed partial decryption. By the evaluation correctness of dTFHE, the output of  $Q$  is computationally indistinguishable from  $\text{hyb}_2$ .

- $\text{hyb}_4$ : If  $Q$  receives a valid tuple  $(\{(\llbracket x_j \rrbracket, \pi_j, \sigma_j)\}_{j \in [n-1]}, \llbracket y \rrbracket, \llbracket y : \text{sk}_i \rrbracket, \pi_i^{\text{dec}})$  from party  $P_i$  in Round 2 (if it holds for multiple parties, then pick the smallest  $i$ ), then decrypt  $\llbracket x_c \rrbracket$  by dTFHE secret keys to obtain  $\tilde{x}$  and output  $f(x_1, \dots, x_{c-1}, \tilde{x}, x_{c+1}, \dots, x_{n-1})$ .

There are two cases of the chosen party  $P_i$ : (a)  $P_i$  is honest, and (b)  $P_i$  is corrupted.

- If  $P_i$  is honest, then  $\{\llbracket x_j \rrbracket\}_{j \in [n-1] \setminus \{c\}}$  are correctly generated ciphertexts of  $\{x_j\}_{j \in [n-1] \setminus \{c\}}$ .
- If  $P_i$  is corrupted, namely  $P_i = P_c$ , then we claim that for each  $j \in [n-1] \setminus \{c\}$ ,  $\llbracket x_j \rrbracket$  is the same as what  $P_j$  sends to  $P_c$  in Round 1. If they are different for any  $j^*$ , then we can use the adversary  $\mathcal{A}$  to forge a signature for  $P_{j^*}$ .

In particular, we construct a PPT  $\mathcal{B}$  to break the unforgeability of the signature scheme as follows.  $\mathcal{B}$  first gets a verification key  $\text{vkey}$  from the challenger. Then  $\mathcal{B}$  generates the simulated CRS  $\text{simcrs}$  and all the public and secret keys except  $(\text{vkey}_{j^*}, \text{skey}_{j^*})$ , and sets  $\text{vkey}_{j^*} := \text{vkey}$ .  $\mathcal{B}$  invokes  $\mathcal{A}$  with  $\text{simcrs}$ , public keys  $(\text{pk}, \{\text{vkey}_i\}_{i \in [n]})$ , and secret keys  $(\text{sk}_c, r_c, \text{skey}_c)$ .  $\mathcal{B}$  follows the protocol as in  $\text{hyb}_4$  except that when computing  $\sigma_{j^*}$  in Round 1, it queries the challenger on message  $(\llbracket x_{j^*} \rrbracket, \pi_{j^*})$  to obtain a signature  $\sigma_{j^*}$ .

Next in Round 2,  $\mathcal{B}$  receives a valid tuple  $(\llbracket x_{j^*} \rrbracket', \pi'_{j^*}, \sigma'_{j^*})$  from  $\mathcal{A}$  where  $\llbracket x_{j^*} \rrbracket' \neq \llbracket x_{j^*} \rrbracket$ .  $\mathcal{B}$  can then output the forged signature  $\sigma'_{j^*}$  on message  $(\llbracket x_{j^*} \rrbracket', \pi'_{j^*})$ .

By the unforgeability of the signature scheme,  $\llbracket x_j \rrbracket$  is the same as what  $P_j$  sends to  $P_c$  in Round 1 for all  $j \in [n-1] \setminus \{c\}$ , hence  $\{\llbracket x_j \rrbracket\}_{j \in [n-1] \setminus \{c\}}$  are correctly generated ciphertexts of  $\{x_j\}_{j \in [n-1] \setminus \{c\}}$ .

By the correctness of dTFHE,  $\{x_j\}_{j \in [n-1] \setminus \{c\}}$  are computationally indistinguishable from  $\{\tilde{x}_j\}_{j \in [n-1] \setminus \{c\}}$  computed in  $\text{hyb}_3$ . Thus  $Q$ 's output in this hybrid is computationally indistinguishable from its output in  $\text{hyb}_3$ .

- $\text{hyb}_5$ : If the party  $P_i$  picked in  $\text{hyb}_4$  is an honest party, then  $P_i$  must be the party with the smallest index that receives a valid tuple  $(\llbracket x_c \rrbracket, \pi_c, \sigma_c)$  from  $\mathcal{A}$  in Round 1. Decrypt  $\llbracket x_c \rrbracket$  by dTFHE secret keys to obtain  $\tilde{x}$  and output  $f(x_1, \dots, x_{c-1}, \tilde{x}, x_{c+1}, \dots, x_{n-1})$ .

This hybrid is identical to  $\text{hyb}_4$  because an honest  $P_i$  forwards  $(\llbracket x_c \rrbracket, \pi_c, \sigma_c)$  to  $Q$  in Round 2.

- $\text{hyb}_6$ : If  $Q$  doesn't receive a valid tuple  $(\{(\llbracket x_j \rrbracket, \pi_j, \sigma_j)\}_{j \in [n-1]}, \llbracket y \rrbracket, \llbracket y : \text{sk}_i \rrbracket, \pi_i^{\text{dec}})$  from any party  $P_i$  in Round 2, then it outputs  $f(x_1, \dots, x_{c-1}, \tilde{x}, x_{c+1}, \dots, x_{n-1})$  if  $\mathcal{A}$  sends  $\tilde{x}$  to  $Q$  in Round 2, or  $f(x_1, \dots, x_{c-1}, \hat{x}_c, x_{c+1}, \dots, x_{n-1})$  otherwise, where  $\hat{x}_c$  is the default input of party  $P_c$ .

This hybrid is identical to  $\text{hyb}_5$  because an honest party either sends a valid tuple in Round 2, or sends its input in the clear. If  $Q$  doesn't receive any valid tuple, then it must have received all the honest parties' inputs and will compute the output accordingly.

- $\text{hyb}_7$ : In Round 1, for each  $i \in [n-1] \setminus \{c\}$ , replace  $\llbracket x_i \rrbracket$  sent to  $\mathcal{A}$  by  $\llbracket 0_i \rrbracket$ .

By the semantic security of dTFHE,  $\mathcal{A}$  cannot distinguish between  $\text{hyb}_6$  and  $\text{hyb}_7$ .

The last hybrid gives the output of Sim and  $Q$  in the ideal world.

**Case 2:  $Q$  is corrupted.** Sim directly receives an output  $y$  from the ideal functionality (recall that  $Q$  has no input). The strategy of Sim is described as follows:

- For each  $i \in [n-1]$ , compute the following:
  - $\llbracket 0_i \rrbracket \leftarrow \text{dTFHE.Enc}(\text{pk}, 0_i)$ .
  - $\pi_i \leftarrow \text{NIZK.Sim.Prove}(\text{td}, \text{st}_i)$  for  $\text{st}_i \in \mathsf{L}_1$  where  $\text{st}_i = (\llbracket 0_i \rrbracket, \text{pk})$ .
  - $\sigma_i \leftarrow \text{Sign}(\text{skey}_i, (\llbracket 0_i \rrbracket, \pi_i))$ .
- Compute  $\llbracket \tilde{y} \rrbracket \leftarrow \text{dTFHE.Eval}(\text{pk}, f, \llbracket 0_1 \rrbracket, \dots, \llbracket 0_{n-1} \rrbracket)$ .
- Compute  $\{\llbracket \tilde{y} : \text{sk}_i \rrbracket\}_{i \in [n-1]} \leftarrow \text{dTFHE.Sim}(f, y, \llbracket 0_1 \rrbracket, \dots, \llbracket 0_{n-1} \rrbracket, \text{sk}_n)$ .
- For each  $i \in [n-1]$ , do the following:
  - Compute  $\pi_i^{\text{dec}} \leftarrow \text{NIZK.Sim.Prove}(\text{td}, \text{st}_i^{\text{dec}})$  for  $\text{st}_i^{\text{dec}} \in \mathsf{L}_2$  where  $\text{st}_i^{\text{dec}} = (\llbracket \tilde{y} : \text{sk}_i \rrbracket, \llbracket \tilde{y} \rrbracket, \text{pk}_i, i)$ .
  - Send  $(\{(\llbracket 0_j \rrbracket, \pi_j, \sigma_j)\}_{j \in [n-1]}, \llbracket \tilde{y} \rrbracket, \llbracket \tilde{y} : \text{sk}_i \rrbracket, \pi_i^{\text{dec}})$  to  $\mathcal{A}$  on behalf of party  $P_i$  in Round 2.
- Finally, output whatever  $\mathcal{A}$  outputs.

**Hybrid argument.** Now we show that  $\mathcal{A}$ 's output in the real world is computationally indistinguishable from Sim's output in the ideal world.

- $\text{hyb}_0$ :  $\mathcal{A}$ 's output in the real world.
- $\text{hyb}_1$ : First generate  $(\text{simcrs}, \text{td}) \leftarrow \text{NIZK.Sim.Setup}(1^\lambda)$  and give  $\text{simcrs}$  to  $\mathcal{A}$  as the CRS. For each  $i \in [n - 1]$ , replace  $\pi_i$  by a simulated NIZK argument  $\text{NIZK.Sim.Prove}(\text{td}, \text{st}_i)$  and  $\pi_i^{\text{dec}}$  by  $\text{NIZK.Sim.Prove}(\text{td}, \text{st}_i^{\text{dec}})$ .  
By the zero-knowledge property of NIZK,  $\mathcal{A}$  cannot distinguish between  $\text{hyb}_0$  and  $\text{hyb}_1$ .
- $\text{hyb}_2$ : Compute  $\{\llbracket y : \text{sk}_i \rrbracket\}_{i \in [n-1]}$  by  $\text{dTFHE.Sim}(f, y, \llbracket x_1 \rrbracket, \dots, \llbracket x_{n-1} \rrbracket, \text{sk}_n)$ .  
By the simulation security of dTFHE, this hybrid is computationally indistinguishable from  $\text{hyb}_1$ .
- $\text{hyb}_3$ : Sim's output in the ideal world. The only difference between this hybrid and  $\text{hyb}_2$  is that  $\llbracket x_i \rrbracket$  is replaced by  $\llbracket 0_i \rrbracket$  for each  $i \in [n - 1]$ . Because of the semantic security of dTFHE,  $\text{hyb}_2$  and  $\text{hyb}_3$  are computationally indistinguishable to  $\mathcal{A}$ .