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

Saikrishna Badrinarayanan[1](✉), Peihan Miao[2], Pratyay Mukherjee[3], and Divya Ravi[4]

[1] LinkedIn, Mountain View, USA
bsaikrishna7393@gmail.com
[2] Brown University, Providence, USA
[3] Supra Research, Kolkata, India
[4] Aarhus University, Aarhus, Denmark

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 the round complexity of fully secure solitary MPC in the honest majority setting and with computational security. 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 to design 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

© International Association for Cryptologic Research 2023
G. Rothblum and H. Wee (Eds.): TCC 2023, LNCS 14370, pp. 124–155, 2023.
https://doi.org/10.1007/978-3-031-48618-0_5

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.

1 Introduction

Secure multiparty computation (MPC) [25,39] 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 [13] 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. [29] 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 [35] 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 [8,9]. 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 [13] does not rule out solitary MPC with god in the presence of a dishonest majority,[1] Halevi et al. [29] showed that there exist functionalities for which solitary MPC with god is also impossible with dishonest majority. Hence, the results of [13] and [29] 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 [4,27,29]). Both impossibility results hold even when

[1] 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.

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 the full version [7] for a detailed literature survey). In the honest majority setting, *three* rounds are known to be necessary [24,28,36] 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 [3,6,28]. The protocol of Gordon et al. [28] requires a CRS[2], while the other two [3,6] 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 [2,21] show that either a broadcast channel or a public key infrastructure (PKI) setup is indeed necessary assuming an honest majority (in particular, when $n/3 \le t < n/2$ for an n-party protocol against adversaries corrupting up to t parties) even with a CRS.[3] Note that although PKI setup and broadcast channels are equivalent according to [17] 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 $n/3 \le t < n/2$ 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 [28,30] 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?*

[2] This protocol uses a decentralized threshold fully homomorphic encryption (dTFHE) scheme. The public parameter of this dTFHE is assumed to be shared among the parties and viewed as a common reference string (refer to [28] for further details).

[3] Fitzi et al. [21] show that converge-cast cannot be achieved when $n/3 \le t < n/2$ in the *information theoretic* setting. Alon et al. [2] 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 $n/3 \le t < n/2$. Both arguments also work even in the presence of a CRS. We present the proof in the full version [7] for completeness.

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 [17] and Fischer and Lynch [19] 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.[4] This can be overcome by considering *randomized* broadcast protocols in the honest majority setting [1,18,20,32] requiring expected constant rounds. In particular, the most round-efficient protocol to our knowledge is proposed by Abraham et al. [1], 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.[5] In fact, it is shown that termination cannot be guaranteed in constant rounds [12,31].

Recent works [14–16,22] 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.

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:

[4] 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) [33,37].

[5] 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. [1] 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 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?*

We answer this question in the affirmative by constructing a *five-round* solitary MPC protocol that achieves god in the above 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 [3,6,28] 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 [36], 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 [3,6,28], 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. [14] identifies which rounds of broadcast are necessary for achieving round-optimal (two-round) standard MPC with *dishonest majority*. The recent work of [15] 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, 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 [6,28] still work if the third-round messages are sent over pairwise-private channels (we defer the discussion to the full version [7]), 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. [28] and uses a decentralized threshold fully homomorphic encryption (dTFHE) scheme (defined in [10]) 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 [28])[6].

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 setting (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

[6] We leave it as an interesting open problem to achieve the upper bound using weaker forms of PKI setup and studying the minimal assumption required.

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 [28] and [17].

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 [30] 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 [28] and [17].

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.

Table 1. Round complexity of solitary MPC with god. "—" means it doesn't matter what value to take. Our results are highlighted in red.

broadcast	PKI	(n, t)	Q has input	lower bound	upper bound
yes	yes	$t < n/2$	—	2 [30]	2 [28]
yes	no	$n/3 \leq t < n/2$	—	3 (Theorem 1)	3 [3,6,28]
no	yes	$n = 3, t = 1$	no	2 [30]	2 (full version [7])
no	yes	$n = 3, t = 1$	yes	3 (full version [7])	3 [28] + [17]
no	yes	$n = 5, t = 2$	—	3 (full version [7])	4 [28] + [17]
no	yes	$3\lceil n/7 \rceil \leq t < n/2$	—	4 (Theorem 4)	5 (Theorem 5)

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.

bc in R1	bc in R2	bc in R3	(n, t)	Possible?
no	yes	yes	$n/3 \leq t < n/2$	No (Theorem 2)
yes	no	yes	$2\lceil n/5 \rceil \leq t < n/2$	No (Theorem 3)
yes	yes	no	$t < n/2$	Yes [3,6,28]
yes	no	no	$n = 3, t = 1$	Yes [36]

1.2 Roadmap

We provide a technical overview in Sect. 2 and preliminaries in Sect. 3. In Sect. 4 we present our lower bound results assuming a broadcast channel but no PKI setup. In Sect. 5 we provide our lower bounds for PKI without broadcast as well as our main five-round protocol as an upper bound. We defer the results for the special cases of $t = 1$ and $t = 2$ to the full version [7].

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. [28] 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 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 pk and individual secret keys sk_i for each party P_i. In Round 1, each party P_i computes an encryption of its input x_i and broadcasts $[\![x_i]\!]$.[7] Then each party can homomorphically evaluate the function f on $[\![x_1]\!], \ldots, [\![x_n]\!]$ to obtain an encryption of the output $[\![y]\!]$. In Round 2, each party broadcasts a partial decryption of $[\![y]\!]$. At the end of this, every party can individually combine the partial decryptions to learn the output y.

[7] We use $[\![x]\!]$ to denote a dTFHE encryption of x.

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 [17], 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 $[\![x_1]\!], \ldots, [\![x_n]\!]$ so that they can homomorphically evaluate on the same set of ciphertexts and compute partial decryptions on the same $[\![y]\!]$. This already implies Byzantine agreement, which requires at least $(t+1)$ rounds [17].

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 $[\![x_n]\!]$ when Q is honest. This can be done in two rounds (by augmenting the two-round broadcast with abort protocol of [26] with digital signatures). In Round 1, Q sends $[\![x_n]\!]$ to each party (along with its signature). To ensure Q sends the same ciphertext to everyone, in Round 2, parties 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 $[\![x_n]\!]$ 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 $[\![x_i]\!]$ 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 $[\![x_i]\!]$ to Q. Then in Round 2, Q sends $[\![x_1]\!], \ldots, [\![x_n]\!]$ 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 $[\![x_i]\!]$; 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: 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 $[\![x_i]\!]$ to every other party. Then in Round 2, each party P_i, besides sending $[\![x_i]\!]$ to Q, also sends all the ciphertexts he has received to Q. In Round 3, Q picks a set of ciphertexts $[\![x_1]\!], \ldots, [\![x_n]\!]$ 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 $[\![x_k]\!]$ 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 $[\![x'_k]\!]$ (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 $[\![x_k]\!]$, 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 [28] is to apply a generic round-preserving compiler [5] 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 Sect. 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. [28] 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 [17] 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. The details are deferred to the full version [7].

2.2 Overview of Lower Bounds

For each of our lower bound proofs, we design a special solitary function f that cannot be computed with god. At a high level, we assume towards a contradiction that there exists an MPC protocol Π that can compute f with god. Next, we analyze a sequence of scenarios which lead us to the final contradiction regarding the properties that Π must satisfy. Here, we exploit the guarantees of correctness, privacy and full-security (guaranteed output delivery). We carefully design the function f and scenarios for each lower bound proof. For certain proofs, we leverage a delicate *probabilistic argument* technique, which we elaborate below.

With Broadcast and no PKI. For our three-round lower bound with a broadcast channel and no PKI setup, we design a solitary function $f(x_1, x_2, x_3)$ among parties P_1, P_2, and Q (output receiving party) that has an oblivious transfer flavor. The function is defined as $f(x_1 = (m_0, m_1), x_2 = b, x_3 = \bot) := m_b$, where $x_3 = \bot$ 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 assume there exists a two-round protocol Π that computes f with god and consider three scenarios. The first scenario involves a malicious P_2 who drops his private message towards Q in Round 1 and aborts in Round 2. The second scenario involves a passive Q who behaves honestly but recomputes the output by locally emulating Scenario 1 in her head. The security guarantee of god provided by Π allow us to argue 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. 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, our specially designed function f allows P_1 to derive P_2's input. We present the full proof in Sect. 4.1.

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 use the same function f as above and assume there exists a three-round protocol Π that computes f with god and 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 Π. 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 and locally compute $f(x_1, x_2, \perp)$. Our specially designed f allows P_1 to derive honest P_2's input, violating the security of Π. A more detailed proof is presented in Sect. 4.2.

Necessity of Broadcast in Round 2. For our result showing necessity of broadcast in Round 2, we design a more sophisticated function f (see Sect. 4.3 for the construction) and leverage a more involved probabilistic argument in our proof. We assume there exists a three-round 5-party solitary MPC Π that computes f 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)** Π 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 Π), then there exists some x_1^* such that the output obtained by Q is *not* computed with respect to x_1^* *with a sufficiently large (constant) 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 Π is such that $\{P_2, Q\}$ obtain sufficient information to compute on x_1 at the end of Round 1 itself. This would make Π susceptible to residual function attack by $\{P_2, Q\}$ which violates security. **(2)** Π 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 Π. 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.

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 Π with god computing a specially designed 7-party solitary function f (see Sect. 5.1 for the construction of f). 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, there must exist some x_1^* such that the output obtained by Q is *not* computed with respect to x_1^* *with a sufficiently large (constant) probability.* Intuitively, this holds because the communication in Scenario 1 is independent of P_1's input. Next, we prove via a sequence of hybrids that in Scenario 2, there also exists x_1^* such that the output is *not* computed on x_1^* *with a sufficiently large probability.* This lets us infer a critical property satisfied by Π - if $\{P_3, P_4, P_5\}$ do not receive any communication *directly* from P_1 throughout Π 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 $x_1'(\neq x_1^*)$ *with a sufficiently large 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 Π 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 $\{P_1, \ldots, P_6\}$ 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^*. 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 some $x_1' \neq x_1^*$ by using the communication from $\{P_1, P_2, P_6\}$ and $\{P_3, P_4, P_5\}$ *selectively*, violating the security of Π.

Finally, 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 full version [7] for details. Notably, all the lower bounds also extend to arbitrary correlated randomness setup.

3 Preliminaries

3.1 Notation and Setting

We use λ to denote the security parameter. By $\mathsf{poly}(\lambda)$ we denote a polynomial function in λ. By $\mathsf{negl}(\lambda)$ 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 λ. We use $[\![x]\!]$ to denote an encryption of x.

We consider a set of parties $\{P_1 \ldots, P_n\}$. 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.

The security definition of solitary MPC with guaranteed output delivery is deferred to the full version.

3.2 Cryptographic Primitives

In our constructions, we need to use digital signatures, simulation-extractible non-interactive zero-knowledge (NIZK) arguments, and decentralized threshold fully homomorphic encryption (dTFHE). In this section, we only define the syntax of dTFHE and the NIZK languages used in our constructions, and defer their security definitions to the full version.

Syntax of dTFHE. We define a t-out-of-n decentralized threshold fully homomorphic encryption scheme with the following syntax as in [10].

Definition 1 ***(Decentralized Threshold Fully Homomorphic Encryption (dTFHE)).*** *Let* $\mathcal{P} = \{P_1, \ldots, P_n\}$ *be a set of parties. A dTFHE scheme is a tuple of PPT algorithms* $\mathsf{dTFHE} = (\mathsf{dTFHE.DistGen}, \mathsf{dTFHE.Enc}, \mathsf{dTFHE.PartialDec}, \mathsf{dTFHE.Eval}, \mathsf{dTFHE.Combine})$ *with the following syntax:*

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

NIZK Languages Used. In our solitary MPC protocols, we will consider two NP languages $\mathsf{L}_1, \mathsf{L}_2$ for the NIZK described below.

- **NP Language L_1:**
  Statement $\mathsf{st} = ([\![x]\!], \mathsf{pk})$ Witness $\mathsf{wit} = (x, \rho)$
  $\mathsf{R}_1(\mathsf{st}, \mathsf{wit}) = 1$ iff $[\![x]\!] = \mathsf{dTFHE.Enc}(\mathsf{pk}, x; \rho)$.
- **NP Language L_2:**
  Statement $\mathsf{st} = ([\![x : \mathsf{sk}]\!], [\![x]\!], \mathsf{pk}, i)$ Witness $\mathsf{wit} = (\mathsf{sk}, r)$
  $\mathsf{R}_2(\mathsf{st}, \mathsf{wit}) = 1$ iff $[\![x : \mathsf{sk}]\!] = \mathsf{dTFHE.PartialDec}(\mathsf{sk}, [\![x]\!])$ and
  $(\mathsf{pk}, \mathsf{sk}) = \mathsf{dTFHE.DistGen}(1^\lambda, 1^d, i; r)$.

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 Sect. 4.1. Then we study whether it is possible to use fewer rounds of broadcast and show in Sect. 4.2 and Sect. 4.3 that broadcast is necessary in both the first and second rounds. The above negative results are tight given the existing results of [3,6,28,36], which we discuss in the full version [7].

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. We discuss in the full version why the existing proofs of lower bounds (three rounds) for standard MPC with god in the presence of an honest majority [24,28,36] do not hold for solitary functionalities.

Theorem 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. Consider 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. We define f as $f(x_1 = (m_0, m_1), x_2 = b, x_3 = \bot) := m_b$, where $x_3 = \bot$ 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. For the sake of contradiction, suppose there exists a two-round 3-party solitary MPC with god, say Π which can compute f. Note that at most the adversary corrupts at most one party.

We consider three different scenarios of the execution of Π. For simplicity, we assume the following about the structure of Π: **(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 [23]. **(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 $\mathsf{pc}_{i \to j}$ denote the pairwise-private communication from P_i to P_j in Round 1 and $\mathsf{b}^r_{i \to}$ 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 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_2 sends messages $\mathsf{pc}_{2 \to 1}, \mathsf{b}^1_{2 \to}$ according to the protocol specification but drops the message $\mathsf{pc}_{2 \to 3}$. 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 $\mathsf{pc}_{2\to3}$ (private communication from P_2 to Q in Round 1) and $\mathsf{b}^2_{2\to}$ (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 $(\mathsf{pc}_{2\to1}, \mathsf{b}^1_{2\to})$ 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 $\widetilde{\mathsf{b}^1_{1\to}}, \widetilde{\mathsf{b}^1_{3\to}}, \widetilde{\mathsf{pc}_{1\to3}}, \widetilde{\mathsf{pc}_{3\to1}}$. Next, locally compute $\widetilde{\mathsf{b}^2_{1\to}}$ using the Round 1 emulated messages which results in the complete view $\widetilde{\mathsf{View}_3}$ of Q analogous to Scenario 1, where $\widetilde{\mathsf{View}_3} = \{\mathsf{crs}, \widetilde{x_3}, \widetilde{r_3}, \widetilde{\mathsf{b}^1_{1\to}}, \mathsf{b}^1_{2\to}, \widetilde{\mathsf{pc}_{1\to3}}, \widetilde{\mathsf{b}^2_{1\to}}\}$ corresponds to the inputs $(\widetilde{x_1}, x_2, \widetilde{x_3})$.

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

	Scenario 1			Scenario 2 & 3		
	View_1	View_2	View_3	View_1	View_2	View_3
Initial Input	(x_1, r_1, crs)	(x_2, r_2, crs)	(x_3, r_3, crs)	(x_1, r_1, crs)	(x_2, r_2, crs)	(x_3, r_3, crs)
Round 1	$\mathsf{pc}_{2\to1}$, $\mathsf{pc}_{3\to1}$ $\mathsf{b}^1_{2\to}, \mathsf{b}^1_{3\to}$	$\mathsf{pc}_{1\to2}$, $\mathsf{pc}_{3\to2}$, $\mathsf{b}^1_{1\to}, \mathsf{b}^1_{3\to}$	$\mathsf{pc}_{1\to3}$, –, $\mathsf{b}^1_{1\to}, \mathsf{b}^1_{2\to}$	$\mathsf{pc}_{2\to1}$, $\mathsf{pc}_{3\to1}$, $\mathsf{b}^1_{2\to}, \mathsf{b}^1_{3\to}$	$\mathsf{pc}_{1\to2}$, $\mathsf{pc}_{3\to2}$, $\mathsf{b}^1_{1\to}, \mathsf{b}^1_{3\to}$	$\mathsf{pc}_{1\to3}$, $\mathsf{pc}_{2\to3}$, $\mathsf{b}^1_{1\to}, \mathsf{b}^1_{2\to}$
Round 2	–	$\mathsf{b}^2_{1\to}$	$\mathsf{b}^2_{1\to}$	$\mathsf{b}^2_{2\to}$	$\mathsf{b}^2_{1\to}$	$\mathsf{b}^2_{1\to}, \mathsf{b}^2_{2\to}$

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 Π is vulnerable to a potential attack by semi-honest Q (that is captured in Scenario 2) which enables Q to learn information that he is not supposed to learn; which 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 Π is such that an adversary corrupting P_1 passively (as in Scenario 3) 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 present the formal proof and show how the proof can be extended for $n \geq 3$ and $n/3 \leq t < n/2$ (using player partitioning technique [34]) in the full version [7].

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 2. *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. Consider the function $f(x_1, x_2, x_3)$ defined as in the proof of Theorem 1, i.e. $f(x_1 = (m_0, m_1), x_2 = b, x_3 = \perp) := m_b$. Suppose for the sake of contradiction that there exists a three-round solitary MPC protocol with god, say Π that computes f and utilizes broadcast channel only in Rounds 2 and 3 (i.e., Π 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 Π 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 [23].

We analyze three different scenarios of the execution of Π. 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 $\mathsf{pc}_{i\to j}$ 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 $\widetilde{\mathsf{pc}_{i\to j}}$ 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 $\mathsf{b}_i^{x,r,\mathsf{pc}_{i-1},\mathsf{pc}_{i+1}}$ 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 pc_{i-1} from P_{i-1} and pc_{i+1} from P_{i+1} in Round 1 (let $P_0 := P_3$ and $P_4 := P_1$). Lastly, let $\mathsf{pc}_{i\to 3}^{\ell}$ where $i \in [2], \ell \in [3]$ denote the pairwise-private communication from P_i to Q in Round 3 in Scenario ℓ. A party's view consists of 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 $\mathsf{pc}_{2\to 1}$ to P_1 and $\widetilde{\mathsf{pc}_{2\to 3}}$ 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 $\mathsf{b}_2^{x_2,r_2,\mathsf{pc}_{1\to 2},\widetilde{\mathsf{pc}_{3\to 2}}}$.
In Round 3, P_2 aborts.

Scenario 2: Adversary corrupts Q. In Round 1, Q behaves towards P_1 using $(x_3 = \bot, r_3)$ while behaving towards P_2 using $(\widetilde{x_3} = \bot, \widetilde{r_3})$. In other words, Q sends $\mathsf{pc}_{3\to1}$ to P_1 and $\widetilde{\mathsf{pc}_{3\to2}}$ to P_2.
In Round 2, Q broadcasts a message as if he behaved honestly in Round 1 to both parties (using $(x_3 = \bot, r_3)$) and received a message from P_2 in Round 1 using $(\widetilde{x_2}, \widetilde{r_2})$. Formally, Q broadcasts $\mathsf{b}_3^{x_3, r_3, \mathsf{pc}_{1\to3}, \widetilde{\mathsf{pc}_{2\to3}}}$.

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

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

	Scenario 1			Scenario 2		
	View_1	View_2	View_3	View_1	View_2	View_3
Initial Input	(x_1, r_1, crs)	(x_2, r_2, crs)	$(x_3 = \bot, r_3, \mathsf{crs})$	(x_1, r_1, crs)	(x_2, r_2, crs)	$(x_3 = \bot, r_3, \mathsf{crs})$
Round 1	$\mathsf{pc}_{2\to1}, \mathsf{pc}_{3\to1}$	$\mathsf{pc}_{1\to2}, \mathsf{pc}_{3\to2}$	$\mathsf{pc}_{1\to3}, \widetilde{\mathsf{pc}_{2\to3}}$	$\mathsf{pc}_{2\to1}, \mathsf{pc}_{3\to1}$	$\mathsf{pc}_{1\to2}, \widetilde{\mathsf{pc}_{3\to2}}$	$\mathsf{pc}_{1\to3}, \mathsf{pc}_{2\to3}$
Round 2	$\mathsf{b}_2^{x_2, r_2, \mathsf{pc}_{1\to2}, \widetilde{\mathsf{pc}_{3\to2}}}$	$\mathsf{b}_1^{x_1, r_1, \mathsf{pc}_{2\to1}, \mathsf{pc}_{3\to1}}$	$\mathsf{b}_1^{x_1, r_1, \mathsf{pc}_{2\to1}, \mathsf{pc}_{3\to1}}$	$\mathsf{b}_2^{x_2, r_2, \mathsf{pc}_{1\to2}, \widetilde{\mathsf{pc}_{3\to2}}}$	$\mathsf{b}_1^{x_1, r_1, \mathsf{pc}_{2\to1}, \mathsf{pc}_{3\to1}}$	$\mathsf{b}_1^{x_1, r_1, \mathsf{pc}_{2\to1}, \mathsf{pc}_{3\to1}}$
	$\mathsf{b}_3^{x_3, r_3, \mathsf{pc}_{1\to3}, \widetilde{\mathsf{pc}_{2\to3}}}$	$\mathsf{b}_3^{x_3, r_3, \mathsf{pc}_{1\to3}, \widetilde{\mathsf{pc}_{2\to3}}}$	$\mathsf{b}_2^{x_2, r_2, \mathsf{pc}_{1\to2}, \widetilde{\mathsf{pc}_{3\to2}}}$	$\mathsf{b}_3^{x_3, r_3, \mathsf{pc}_{1\to3}, \widetilde{\mathsf{pc}_{2\to3}}}$	$\mathsf{b}_3^{x_3, r_3, \mathsf{pc}_{1\to3}, \widetilde{\mathsf{pc}_{2\to3}}}$	$\mathsf{b}_2^{x_2, r_2, \mathsf{pc}_{1\to2}, \widetilde{\mathsf{pc}_{3\to2}}}$
Round 3	–	–	$\mathsf{pc}_{1\to3}^1$	–	–	$\mathsf{pc}_{1\to3}^2, \mathsf{pc}_{2\to3}^2$

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

	View_1	View_2	View_3
Initial Input	(x_1, r_1, crs)	(x_2, r_2, crs)	$(x_3 = \bot, r_3, \mathsf{crs})$
Round 1	$\mathsf{pc}_{2\to1}, \mathsf{pc}_{3\to1}$	$\mathsf{pc}_{1\to2}, \mathsf{pc}_{3\to2}$	$\mathsf{pc}_{1\to3}, \mathsf{pc}_{2\to3}$
Round 2	$\mathsf{b}_2^{x_2, r_2, \mathsf{pc}_{1\to2}, \mathsf{pc}_{3\to2}}$	$\mathsf{b}_1^{x_1, r_1, \mathsf{pc}_{2\to1}, \mathsf{pc}_{3\to1}}$	$\mathsf{b}_1^{x_1, r_1, \mathsf{pc}_{2\to1}, \mathsf{pc}_{3\to1}}$
	$\mathsf{b}_3^{x_3, r_3, \mathsf{pc}_{1\to3}, \mathsf{pc}_{2\to3}}$	$\mathsf{b}_3^{x_3, r_3, \mathsf{pc}_{1\to3}, \mathsf{pc}_{2\to3}}$	$\mathsf{b}_2^{x_2, r_2, \mathsf{pc}_{1\to2}, \mathsf{pc}_{3\to2}}$
Round 3	–	–	$\mathsf{pc}_{1\to3}^3, \mathsf{pc}_{2\to3}^3$

The proof skeleton is as follows. First, we claim if Scenario 1 occurs, then Q must obtain $f(x_1, x_2, \bot)$ with overwhelming probability. Due to the god property of Π, 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 is 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 $f(x_1, x_2, \bot)$, which compromises the security of Π. We defer the formal proof to the full version [7].

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 (refer full version [7]).

Theorem 3. *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 $2\lceil n/5 \rceil \leq t < n/2$. Consider the solitary function $f(x_1, \ldots, x_5)$ among $\{P_1, \ldots, P_5\}$ 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. First, set $k = 10$ (looking ahead, we set k to be sufficiently large for the probability arguments to go through). Let $f(x_1 = (x_c, x_r), x_2 = (x_2^0, x_2^1), x_3 = (x_3^0, x_3^1), x_4 = \bot, x_5 = \bot)$ be defined as follows, where $x_c \in \{0, 1\}$, $x_r, x_2^0, x_2^1, x_3^0, x_3^1 \in \{0, 1\}^k$ and $x_2^0 \neq x_2^1, x_3^0 \neq x_3^1$:

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

Suppose for the sake of contradiction that there exists a three-round 5-party solitary MPC protocol with god against two corruptions, say Π that computes f and utilizes broadcast channel only in Round 1 and Round 3 (i.e. Π 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 the following structure for Π: **(a)** Round 3 involves only private messages sent to Q - no broadcast messages. 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). **(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)**).

We consider an execution of Π with inputs $(x_1, \ldots, x_5)$ where x_i denotes the input of P_i. In the above definition of f, $x_4 = x_5 = \bot$ 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 b_i^1 denote the broadcast communication by P_i in Round 1 when P_i behaves honestly. In Rounds 1 and 2, let $\mathsf{pc}_{i \to j}^r$ 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 $\widetilde{\mathsf{pc}}_{i \to j}^2$ 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 ℓ indicating the scenario i.e. $\mathsf{pc}^{3,\ell}_{j\to 5}$ denotes P_j's Round 3 message to Q in Scenario ℓ ($j \in [4], \ell \in [4]$). These messages may be a function of the common reference string (denoted by crs). A party's view comprises of 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 6, 7, 8 and 9.

[**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 : b^1_1, $\mathsf{pc}^1_{1\to 2}$ and $\mathsf{pc}^1_{1\to 5}$.

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 $\widetilde{\mathsf{pc}}^2_{i\to 2}$ 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.

Table 6. Views of $\{P_1, \ldots, P_5\}$ in Scenario 1.

	View_1	View_2	View_3	View_4	View_5
Initial Input	(x_1, r_1, crs)	(x_2, r_2, crs)	(x_3, r_3, crs)	(x_4, r_4, crs)	(x_5, r_5, crs)
Round 1	$\{\mathsf{b}^1_j\}_{j\in[5]\setminus\{1\}}$, $\{\mathsf{pc}^1_{j\to 1}\}_{j\in[5]\setminus\{1\}}$	$\{\mathsf{b}^1_j\}_{j\in[5]\setminus\{2\}}$, $\{\mathsf{pc}^1_{j\to 2}\}_{j\in[5]\setminus\{2\}}$	$\{\mathsf{b}^1_j\}_{j\in[5]\setminus\{3\}}$, $\{\mathsf{pc}^1_{j\to 3}\}_{j\in[5]\setminus\{1,3\}}$	$\{\mathsf{b}^1_j\}_{j\in[5]\setminus\{4\}}$, $\{\mathsf{pc}^1_{j\to 4}\}_{j\in[5]\setminus\{1,4\}}$	$\{\mathsf{b}^1_j\}_{j\in[5]\setminus\{5\}}$, $\{\mathsf{pc}^1_{j\to 5}\}_{j\in[5]\setminus\{5\}}$
Round 2	$\{\mathsf{pc}^2_{j\to 1}\}_{j\in\{2,5\}}$ $\{\widetilde{\mathsf{pc}}^2_{j\to 1}\}_{j\in\{3,4\}}$	$\{\mathsf{pc}^2_{j\to 2}\}_{j\in\{5\}}$ $\{\widetilde{\mathsf{pc}}^2_{j\to 2}\}_{j\in\{3,4\}}$	$\{\mathsf{pc}^2_{j\to 3}\}_{j\in\{2,5\}}$ $\{\widetilde{\mathsf{pc}}^2_{j\to 3}\}_{j\in\{4\}}$	$\{\mathsf{pc}^2_{j\to 4}\}_{j\in\{2,5\}}$ $\{\widetilde{\mathsf{pc}}^2_{j\to 4}\}_{j\in\{3\}}$	– –
Round 3	–	–	–	–	$\{\mathsf{pc}^{3,1}_{j\to 5}\}_{j\in\{2,3,4\}}$

Table 7. Views of $\{P_1, \ldots, P_5\}$ in Scenario 2.

	View_1	View_2	View_3	View_4	View_5
Initial Input	(x_1, r_1, crs)	(x_2, r_2, crs)	(x_3, r_3, crs)	(x_4, r_4, crs)	(x_5, r_5, crs)
Round 1	$\{\mathsf{b}_j^1\}_{j\in[5]\setminus\{1\}}$, $\{\mathsf{pc}_{j\to1}^1\}_{j\in[5]\setminus\{1\}}$	$\{\mathsf{b}_j^1\}_{j\in[5]\setminus\{2\}}$, $\{\mathsf{pc}_{j\to2}^1\}_{j\in[5]\setminus\{2\}}$	$\{\mathsf{b}_j^1\}_{j\in[5]\setminus\{3\}}$, $\{\mathsf{pc}_{j\to3}^1\}_{j\in[5]\setminus\{1,3\}}$	$\{\mathsf{b}_j^1\}_{j\in[5]\setminus\{4\}}$, $\{\mathsf{pc}_{j\to4}^1\}_{j\in[5]\setminus\{1,4\}}$	$\{\mathsf{b}_j^1\}_{j\in[5]\setminus\{5\}}$, $\{\mathsf{pc}_{j\to5}^1\}_{j\in[5]\setminus\{5\}}$
Round 2	$\{\mathsf{pc}_{j\to1}^2\}_{j\in\{2,5\}}$ $\{\widetilde{\mathsf{pc}}_{j\to1}^2\}_{j\in\{3,4\}}$	$\{\mathsf{pc}_{j\to2}^2\}_{j\in\{5\}}$ $\{\widetilde{\mathsf{pc}}_{j\to2}^2\}_{j\in\{3,4\}}$	$\{\mathsf{pc}_{j\to3}^2\}_{j\in\{2,5\}}$ $\{\widetilde{\mathsf{pc}}_{j\to3}^2\}_{j\in\{4\}}$	$\{\mathsf{pc}_{j\to4}^2\}_{j\in\{2,5\}}$ $\{\widetilde{\mathsf{pc}}_{j\to4}^2\}_{j\in\{3\}}$	– –
Round 3	–	–	–	–	$\{\mathsf{pc}_{j\to5}^{3,2}\}_{j\in\{3,4\}}$

Table 8. Views of $\{P_1, \ldots, P_5\}$ in Scenario 3.

	View_1	View_2	View_3	View_4	View_5
Initial Input	(x_1, r_1, crs)	(x_2, r_2, crs)	(x_3, r_3, crs)	(x_4, r_4, crs)	(x_5, r_5, crs)
Round 1	$\{\mathsf{b}_j^1\}_{j\in[5]\setminus\{1\}}$, $\{\mathsf{pc}_{j\to1}^1\}_{j\in[5]\setminus\{1\}}$	$\{\mathsf{b}_j^1\}_{j\in[5]\setminus\{2\}}$, $\{\mathsf{pc}_{j\to2}^1\}_{j\in[5]\setminus\{2\}}$	$\{\mathsf{b}_j^1\}_{j\in[5]\setminus\{3\}}$, $\{\mathsf{pc}_{j\to3}^1\}_{j\in[5]\setminus\{3\}}$	$\{\mathsf{b}_j^1\}_{j\in[5]\setminus\{4\}}$, $\{\mathsf{pc}_{j\to4}^1\}_{j\in[5]\setminus\{4\}}$	$\{\mathsf{b}_j^1\}_{j\in[5]\setminus\{5\}}$, $\{\mathsf{pc}_{j\to5}^1\}_{j\in[5]\setminus\{5\}}$
Round 2	$\{\mathsf{pc}_{j\to1}^2\}_{j\in\{2,5\}}$	$\{\mathsf{pc}_{j\to2}^2\}_{j\in\{1,5\}}$, $\{\widetilde{\mathsf{pc}}_{j\to2}^2\}_{j\in\{3,4\}}$	$\{\mathsf{pc}_{j\to3}^2\}_{j\in\{1,2,5\}}$	$\{\mathsf{pc}_{j\to4}^2\}_{j\in\{1,2,5\}}$	–
Round 3	–	–	–	–	$\{\mathsf{pc}_{j\to5}^{3,3}\}_{j\in\{1,2\}}$

Table 9. Views of $\{P_1, \ldots, P_5\}$ in Scenario 4.

	View_1	View_2	View_3	View_4	View_5
Initial Input	(x_1, r_1, crs)	(x_2, r_2, crs)	(x_3, r_3, crs)	(x_4, r_4, crs)	(x_5, r_5, crs)
Round 1	$\{\mathsf{b}_j^1\}_{j\in[5]\setminus\{1\}}$, $\{\mathsf{pc}_{j\to1}^1\}_{j\in[5]\setminus\{1\}}$	$\{\mathsf{b}_j^1\}_{j\in[5]\setminus\{2\}}$, $\{\mathsf{pc}_{j\to2}^1\}_{j\in[5]\setminus\{2\}}$	$\{\mathsf{b}_j^1\}_{j\in[5]\setminus\{3\}}$, $\{\mathsf{pc}_{j\to3}^1\}_{j\in[5]\setminus\{1,3\}}$	$\{\mathsf{b}_j^1\}_{j\in[5]\setminus\{4\}}$, $\{\mathsf{pc}_{j\to4}^1\}_{j\in[5]\setminus\{1,4\}}$	$\{\mathsf{b}_j^1\}_{j\in[5]\setminus\{5\}}$, $\{\mathsf{pc}_{j\to5}^1\}_{j\in[5]\setminus\{5\}}$
Round 2	$\{\mathsf{pc}_{j\to1}^2\}_{j\in\{2,5\}}$ $\{\widetilde{\mathsf{pc}}_{j\to1}^2\}_{j\in\{3,4\}}$	$\{\mathsf{pc}_{j\to2}^2\}_{j\in\{1,5\}}$ $\{\widetilde{\mathsf{pc}}_{j\to2}^2\}_{j\in\{3,4\}}$	$\{\mathsf{pc}_{j\to3}^2\}_{j\in\{2,5\}}$ $\{\widetilde{\mathsf{pc}}_{j\to3}^2\}_{j\in\{4\}}$	$\{\mathsf{pc}_{j\to4}^2\}_{j\in\{2,5\}}$ $\{\widetilde{\mathsf{pc}}_{j\to4}^2\}_{j\in\{3\}}$	– –
Round 3	– –	– –	– –	– –	$\{\mathsf{pc}_{j\to5}^{3,4}\}_{j\in\{1,2\}} = \{\mathsf{pc}_{j\to5}^{3,3}\}_{j\in\{1,2\}}$ $\{\mathsf{pc}_{j\to5}^{3,4}\}_{j\in\{3,4\}} = \{\mathsf{pc}_{j\to5}^{3,2}\}_{j\in\{3,4\}}$

The proof skeleton is as follows. First, we claim that there exists an $x_c^* \in \{0,1\}$ and $x_r^* \in \{0,1\}^k$ such that if Scenario 1 occurs with respect to $x_1 = (x_c^*, x_r^*)$ and uniformly randomly sampled x_2 and x_3, then the output obtained by Q must be computed with respect to $\neg x_c^*$ with a sufficiently large (constant) probability. Intuitively, if for all x_c and x_r, 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 Π vulnerable to a residual function attack by $\{P_2, Q\}$. Next, we claim the same statement also holds for Scenario 2 (with a different probability). Regarding Scenario 3, correctness of Π 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 Π. This completes the proof sketch. We present the formal proof and show its extension to $2\lceil n/5 \rceil \le t < n/2$ in the full version [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.

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 pubic-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 [28]. 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 the full version [7]. The impossibility for the general case is formally stated below.

Theorem 4. *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 a 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 in the full version). Consider the solitary function $f(x_1, , \ldots, x_7)$ among $\{P_1, \ldots, P_7\}$ 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. First, set $k = 10$ (looking ahead, we set k to be sufficiently large for the probability arguments to go through). Let $f(x_1, x_2 = \bot, x_3 = (x_3^0, x_3^1), x_4 = (x_4^0, x_4^1), x_5 = \bot, x_6 = (x_6^0, x_6^1), x_7 = \bot)$ be defined as follows, where $x_1 \in \{0, 1\}$, $x_3^0, x_3^1, x_4^0, x_4^1, x_6^0, x_6^1 \in \{0, 1\}^k$ and $x_3^0 \neq x_3^1, x_4^0 \neq x_4^1, x_6^0 \neq x_6^1$:

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

In the definition, $x_2 = x_5 = x_7 = \perp$ indicates that P_2, P_5, P_7 do not have any inputs. Suppose for the sake of contradiction that there exists a three-round solitary MPC protocol with god, say Π that computes f.

Without loss of generality, we assume that Π has the following structure: **(a)** Round 3 involves only messages sent to Q; **(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 Π with inputs $(x_1, \ldots, x_7)$ 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 $\mathsf{pc}^r_{i \to j}$ 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 $\widetilde{\mathsf{pc}}^2_{i \to j}$ to denote the messages that P_i ($i \in [7]$) is supposed to send in Round 2 to P_j ($j \in [6] \setminus i$) 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 ℓ indicating the scenario i.e. $\mathsf{pc}^{3,\ell}_{j \to 7}$ denotes P_j's Round 3 message to Q in Scenario ℓ ($j \in [6], \ell \in [4]$). These messages may be a function of the common reference string (denoted by crs) and the PKI setup. Let α_i denote the output of the PKI setup (or more generally, the output of an arbitrary correlated randomness setup) to party P_i. A party's view comprises of crs, α_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 $\{P_1, \ldots, P_6\}$ into two sets $\{P_1, P_2, P_6\}$ and $\{P_3, P_4, P_5\}$. Looking ahead, the final scenario 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 such that Scenarios 1 and 2 let us conclude that if P_1 behaves honestly only in its messages to P_6, then there must exist some $x_1^* \in \{0, 1\}$ such that the communication from $\{P_3, P_4, P_5\}$ to Q enables Q to obtain output with respect $\neg x_1^*$ with a sufficiently large 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 scenario, on behalf of the corrupt parties, we assume that the adversary uses the honest input and its malicious behaviour is limited to dropping some of the messages that were received or supposed to be sent. The views of the parties across various scenarios are described in Tables 10, 11, 12 and 13.

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 $\widetilde{\mathsf{pc}}^2_{6\to j}$ 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 $\mathsf{pc}^2_{6\to j}$ 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 $\widetilde{\mathsf{pc}}^2_{i\to 6}$ 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:[8]

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 $\widetilde{\mathsf{pc}}^2_{i\to j}$ 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 $\mathsf{pc}^2_{i\to j}$ 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.

Table 10. Views of $\{P_1 \dots P_7\}$ in Scenario 1.

	View_1	View_2	View_3	View_4	View_5	View_6	View_7
Initial Input	$(x_1, r_1, \mathsf{crs}, \alpha_1)$	$(x_2, r_2, \mathsf{crs}, \alpha_2)$	$(x_3, r_3, \mathsf{crs}, \alpha_3)$	$(x_4, r_4, \mathsf{crs}, \alpha_4)$	$(x_5, r_5, \mathsf{crs}, \alpha_5)$	$(x_6, r_6, \mathsf{crs}, \alpha_6)$	$(x_7, r_7, \mathsf{crs}, \alpha_7)$
Round 1	$\{\mathsf{pc}^1_{j\to 1}\}_{j\in[7]\setminus\{1\}}$	$\{\mathsf{pc}^1_{j\to 2}\}_{j\in[7]\setminus\{1,2\}}$	$\{\mathsf{pc}^1_{j\to 3}\}_{j\in[7]\setminus\{1,3\}}$	$\{\mathsf{pc}^1_{j\to 4}\}_{j\in[7]\setminus\{1,4\}}$	$\{\mathsf{pc}^1_{j\to 5}\}_{j\in[7]\setminus\{1,5\}}$	$\{\mathsf{pc}^1_{j\to 6}\}_{j\in[7]\setminus\{1,6\}}$	$\{\mathsf{pc}^1_{j\to 7}\}_{j\in[7]\setminus\{1,7\}}$
Round 2	$\{\widetilde{\mathsf{pc}}^2_{j\to 1}\}_{j\in[7]\setminus\{1\}}$	$\{\widetilde{\mathsf{pc}}^2_{j\to 2}\}_{j\in[7]\setminus\{1,2\}}$	$\{\widetilde{\mathsf{pc}}^2_{j\to 3}\}_{j\in[7]\setminus\{1,3\}}$	$\{\widetilde{\mathsf{pc}}^2_{j\to 4}\}_{j\in[7]\setminus\{1,4\}}$	$\{\widetilde{\mathsf{pc}}^2_{j\to 5}\}_{j\in[7]\setminus\{1,5\}}$	$\{\widetilde{\mathsf{pc}}^2_{j\to 6}\}_{j\in[7]\setminus\{1,6\}}$	–
Round 3	–	–	–	–	–	–	$\{\mathsf{pc}^{3,1}_{j\to 7}\}_{j\in\{2,3,4,5\}}$

Table 11. Views of $\{P_1 \dots P_7\}$ in Scenario 2.

	View_1	View_2	View_3	View_4	View_5	View_6	View_7
Initial Input	$(x_1, r_1, \mathsf{crs}, \alpha_1)$	$(x_2, r_2, \mathsf{crs}, \alpha_2)$	$(x_3, r_3, \mathsf{crs}, \alpha_3)$	$(x_4, r_4, \mathsf{crs}, \alpha_4)$	$(x_5, r_5, \mathsf{crs}, \alpha_5)$	$(x_6, r_6, \mathsf{crs}, \alpha_6)$	$(x_7, r_7, \mathsf{crs}, \alpha_7)$
Round 1	$\{\mathsf{pc}^1_{j\to 1}\}_{j\in[7]\setminus\{1\}}$	$\{\mathsf{pc}^1_{j\to 2}\}_{j\in[7]\setminus\{1,2\}}$	$\{\mathsf{pc}^1_{j\to 3}\}_{j\in[7]\setminus\{1,3\}}$	$\{\mathsf{pc}^1_{j\to 4}\}_{j\in[7]\setminus\{1,4\}}$	$\{\mathsf{pc}^1_{j\to 5}\}_{j\in[7]\setminus\{1,5\}}$	$\{\mathsf{pc}^1_{j\to 6}\}_{j\in[7]\setminus\{1,6\}}$	$\{\mathsf{pc}^1_{j\to 7}\}_{j\in[7]\setminus\{1,7\}}$
Round 2	$\{\widetilde{\mathsf{pc}}^2_{j\to 1}\}_{j\in\{2,3,4,5,7\}}$ $\mathsf{pc}^2_{6\to 1}$	$\{\widetilde{\mathsf{pc}}^2_{j\to 2}\}_{j\in\{3,4,5,7\}}$ $\mathsf{pc}^2_{6\to 2}$	$\{\widetilde{\mathsf{pc}}^2_{j\to 3}\}_{j\in\{2,4,5,7\}}$ $\mathsf{pc}^2_{6\to 3}$	$\{\widetilde{\mathsf{pc}}^2_{j\to 4}\}_{j\in\{2,3,5,7\}}$ $\mathsf{pc}^2_{6\to 4}$	$\{\widetilde{\mathsf{pc}}^2_{j\to 5}\}_{j\in\{2,3,4,7\}}$ $\mathsf{pc}^2_{6\to 5}$	$\{\widetilde{\mathsf{pc}}^2_{j\to 6}\}_{j\in\{2,3,4,5,7\}}$	–
Round 3	–	–	–	–	–	–	$\{\mathsf{pc}^{3,2}_{j\to 7}\}_{j\in\{2,3,4,5\}}$

[8] Generally, communication between corrupt parties need not be specified but we include it here for easier understanding of Table 13.

Table 12. Views of $\{P_1 \dots P_7\}$ in Scenario 3.

	View_1	View_2	View_3	View_4	View_5	View_6	View_7
Initial Input	$(x_1, r_1, \mathsf{crs}, \alpha_1)$	$(x_2, r_2, \mathsf{crs}, \alpha_2)$	$(x_3, r_3, \mathsf{crs}, \alpha_3)$	$(x_4, r_4, \mathsf{crs}, \alpha_4)$	$(x_5, r_5, \mathsf{crs}, \alpha_5)$	$(x_6, r_6, \mathsf{crs}, \alpha_6)$	$(x_7, r_7, \mathsf{crs}, \alpha_7)$
Round 1	$\{\mathsf{pc}^1_{j\to1}\}_{j\in[7]\setminus\{1\}}$	$\{\mathsf{pc}^1_{j\to2}\}_{j\in[7]\setminus\{2\}}$	$\{\mathsf{pc}^1_{j\to3}\}_{j\in[7]\setminus\{3\}}$	$\{\mathsf{pc}^1_{j\to4}\}_{j\in[7]\setminus\{4\}}$	$\{\mathsf{pc}^1_{j\to5}\}_{j\in[7]\setminus\{5\}}$	$\{\mathsf{pc}^1_{j\to6}\}_{j\in[7]\setminus\{6\}}$	$\{\mathsf{pc}^1_{j\to7}\}_{j\in[7]\setminus\{7\}}$
Round 2	$\{\mathsf{pc}^2_{j\to1}\}_{j\in\{2,6,7\}}$	$\{\mathsf{pc}^2_{j\to2}\}_{j\in\{1,6,7\}}$	$\{\mathsf{pc}^2_{j\to3}\}_{j\in\{1,2,6,7\}}$	$\{\mathsf{pc}^2_{j\to4}\}_{j\in\{1,2,6.7\}}$	$\{\mathsf{pc}^2_{j\to5}\}_{j\in\{1,2,6.7\}}$	$\{\mathsf{pc}^2_{j\to6}\}_{j\in\{1,2,7\}}$ $\{\widetilde{\mathsf{pc}}^2_{j\to6}\}_{j\in\{3,4,5\}}$	–
Round 3	–	–	–	–	–	–	$\{\mathsf{pc}^{3,3}_{j\to7}\}_{j\in\{1,2,6\}}$

Table 13. Views of $\{P_1 \dots P_7\}$ in Scenario 4.

	View_1	View_2	View_3	View_4	View_5	View_6	View_7
Initial Input	$(x_1, r_1, \mathsf{crs}, \alpha_1)$	$(x_2, r_2, \mathsf{crs}, \alpha_2)$	$(x_3, r_3, \mathsf{crs}, \alpha_3)$	$(x_4, r_4, \mathsf{crs}, \alpha_4)$	$(x_5, r_5, \mathsf{crs}, \alpha_5)$	$(x_6, r_6, \mathsf{crs}, \alpha_6)$	$(x_7, r_7, \mathsf{crs}, \alpha_7)$
Round 1	$\{\utilde{\mathsf{pc}}^1_{j\to1}\}_{j\in[7]\setminus\{1\}}$	$\{\utilde{\mathsf{pc}}^1_{j\to2}\}_{j\in[7]\setminus\{2\}}$	$\{\utilde{\mathsf{pc}}^1_{j\to3}\}_{j\in[7]\setminus\{1,3\}}$	$\{\utilde{\mathsf{pc}}^1_{j\to4}\}_{j\in[7]\setminus\{1,4\}}$	$\{\utilde{\mathsf{pc}}^1_{j\to5}\}_{j\in[7]\setminus\{1,5\}}$	$\{\utilde{\mathsf{pc}}^1_{j\to6}\}_{j\in[7]\setminus\{6\}}$	$\{\mathsf{pc}^1_{j\to7}\}_{j\in[7]\setminus\{7\}}$
Round 2	$\{\mathsf{pc}^2_{j\to1}\}_{j\in\{3,4,5\}}$ $\{\mathsf{pc}^2_{j\to1}\}_{j\in\{2,6,7\}}$	$\{\mathsf{pc}^2_{j\to2}\}_{j\in\{3,4,5\}}$ $\{\mathsf{pc}^2_{j\to2}\}_{j\in\{1,6,7\}}$	$\{\mathsf{pc}^2_{j\to3}\}_{j\in\{2,4,5,7\}}$ $\mathsf{pc}^2_{6\to3}$	$\{\mathsf{pc}^2_{j\to4}\}_{j\in\{2,3,5,7\}}$ $\mathsf{pc}^2_{6\to4}$	$\{\mathsf{pc}^2_{j\to5}\}_{\{2,3,4,7\}}$ $\mathsf{pc}^2_{6\to5}$	$\{\mathsf{pc}^2_{j\to6}\}_{j\in\{3,4,5\}}$ $\{\mathsf{pc}^2_{j\to6}\}_{j\in\{1,2,7\}}$	–
Round 3	–	–	–	–	–	–	$\{\mathsf{pc}^{3,4}_{j\to7} \equiv \mathsf{pc}^{3,3}_{j\to7}\}_{j\in\{1,2,6\}}$ $\{\mathsf{pc}^{3,4}_{j\to7} \equiv \mathsf{pc}^{3,2}_{j\to6}\}_{j\in\{3,4,5\}}$

The proof outline is as follows. First, we show that there exits $x_1^* \in \{0, 1\}$ such that if Scenario 1 occurs with respect to x_1^* and uniformly randomly sampled x_3, x_4, x_6, then the output obtained by Q is computed on $\neg x_1^*$ with a sufficiently large (constant) probability. Next, we show this is also the case for Scenario 2 (with a different probability). 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. We define a sequence of hybrids $\mathsf{hyb}_0, \dots, \mathsf{hyb}_4$. Specifically, hyb_0 and hyb_4 refer to Scenario 1 and 2 respectively and hyb_i is same as 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. $\neg x_1^*$ with a sufficiently large (but slightly different) 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 Π. Lastly, we show that such a protocol Π 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 $\neg x_1^*$), which is a contradiction to security of Π. We defer the formal proof to the full version [7].

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 $\mathsf{dTFHE} = (\mathsf{dTFHE.DistGen}$, $\mathsf{dTFHE.Enc}, \mathsf{dTFHE.PartialDec}, \mathsf{dTFHE.Eval}, \mathsf{dTFHE.Combine})$, a digital signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, and a simulation-extractible NIZK argument

$(\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$. We use the NIZK argument for two NP languages $\mathsf{L}_1, \mathsf{L}_2$ defined in Sect. 3.2. All of them can be built assuming LWE [10,11,38]. Formally, we show the following theorem:

Theorem 5. *Assuming LWE, protocol $\Pi_{5-\mathsf{round}}$ 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 $P_1, \ldots, P_n$ 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 $\mathsf{msg}^{i \to j}$ 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 $[\![x_i]\!]$ along with a NIZK argument π_i proving that the encryption is well formed. On top of that, P_i also attaches its signature $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{skey}_i, ([\![x_i]\!], \pi_i))$. In Round 2, each party sends all the messages it received in Round 1 to Q. In Round 3, Q first initializes a string $\mathsf{msg} = \bot$ 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 σ_i and the NIZK π_i verify successfully) it includes the message in msg and sets a value $\mathsf{ct}_i = [\![x_i]\!]$. Else, in Round 2, if a different party P_{i_1}, forwards a *valid* message $([\![x_i]\!]^{i_1 \to n}, \pi^{i_1 \to n}, \sigma^{i_1 \to n})$ received from P_i in Round 1, include that in msg and set ct_i to be $[\![x_i]\!]^{i_1 \to n}$. If no such i_1 exists, set $\mathsf{ct}_i = \bot$ and append $\bot$ to msg. Then, Q sends msg and a signature on it σ_{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 $[\![y]\!] = \mathsf{dTFHE.Eval}(f, \mathsf{ct}_1, \ldots, \mathsf{ct}_n)$ if: (i) in Round 3, Q sent $(\mathsf{msg}, \sigma_{\mathsf{msg}})$ such that σ_{msg} verifies, (ii) it did not receive a different tuple $(\mathsf{msg}', \sigma_{\mathsf{msg}'})$ from another party in Round 4 such that $\sigma_{\mathsf{msg}'}$ verifies, (iii) in the string msg, every tuple of the form $([\![x_j]\!], \pi_j, \sigma_j)$ 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 msg, there must exist *some valid* tuple of the form $([\![x'_k]\!], \pi'_k, \sigma'_k)$ 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 the full version [7].

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

PKI Setup:

– For each $i \in [n]$: sample $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{dTFHE.DistGen}(1^\lambda, 1^d, i; r_i)$ and $(\mathsf{vkey}_i, \mathsf{skey}_i) \leftarrow \mathsf{Gen}(1^\lambda)$.
– Public key: $\mathsf{pk} = \mathsf{pk}_1 \| \ldots \| \mathsf{pk}_n$ and $\{\mathsf{vkey}_i\}_{i \in [n]}$.
– Secret keys: $(\mathsf{sk}_i, r_i, \mathsf{skey}_i)$ 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 $[\![x_i]\!] \leftarrow \mathsf{dTFHE.Enc}(\mathsf{pk}, x_i; \rho_i)$ using randomness ρ_i, $\pi_i \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{st}_i, \mathsf{wit}_i)$ for $\mathsf{st}_i \in \mathsf{L}_1$ where $\mathsf{st}_i = ([\![x_i]\!], \mathsf{pk})$ and $\mathsf{wit}_i = (x_i, \rho_i)$.
   - Then, compute $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{skey}_i, ([\![x_i]\!], \pi_i))$ and send $([\![x_i]\!], \pi_i, \sigma_i)$ 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 $\mathsf{msg}, \mathsf{ct}_1, \ldots, \mathsf{ct}_n$ as $\perp$.
   - For each $i \in [n]$, let $\{([\![x_j]\!]^{i \to n}, \pi_j^{i \to n}, \sigma_j^{i \to n})\}_{j \in [n] \setminus \{i\}}$ denote the message received from P_i in Round 2 and $([\![x_i]\!]^{i \to n}, \pi_i^{i \to n}, \sigma_i^{i \to n})$ denote the message received from P_i in Round 1.
   - For each $j \in [n]$, do the following:
     - Let $\{([\![x_j]\!]^{1 \to n}, \pi_j^{1 \to n}, \sigma_j^{1 \to n}), \ldots, ([\![x_j]\!]^{n \to n}, \pi_j^{n \to n}, \sigma_j^{n \to n})\}$ be the messages received across both rounds on behalf of party P_j.
     - Pick the lowest i_1 such that $\mathsf{Verify}(\mathsf{vkey}_j, ([\![x_j]\!]^{i_1 \to n}, \pi_j^{i_1 \to n}), \sigma_j^{i_1 \to n}) = 1$ and $\mathsf{NIZK.Verify}(\mathsf{crs}, \pi_j^{i_1 \to n}, \mathsf{st}_j) = 1$ for $\mathsf{st}_j \in \mathsf{L}_1$ where $\mathsf{st}_j = ([\![x_j]\!]^{i_1 \to n}, \mathsf{pk})$. Set $\mathsf{ct}_j := [\![x_j]\!]^{i_1 \to n}$ and $\mathsf{msg} := \mathsf{msg} \| \text{“}\mathsf{Party\ j}\text{ ”} \| ([\![x_j]\!]^{i_1 \to n}, \pi_j^{i_1 \to n}, \sigma_j^{i_1 \to n})$.
     - If no such i_1 exists, set $\mathsf{msg} = \mathsf{msg} \| \text{“}\mathsf{Party\ j}\text{ ”} \| \perp$.
   - Compute $\sigma_{\mathsf{msg}} \leftarrow \mathsf{Sign}(\mathsf{skey}_n, \mathsf{msg})$. Send $(\mathsf{msg}, \sigma_{\mathsf{msg}})$ to all parties.
   - Set $[\![y]\!] = \mathsf{dTFHE.Eval}(\mathsf{pk}, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_n)$.[9]
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 $\{(\mathsf{msg}^{j \to i}, \sigma_{\mathsf{msg}}^{j \to i})\}_{j \in [n-1] \setminus \{i\}}$ be the messages received in Round 4 and $(\mathsf{msg}^{n \to i}, \sigma_{\mathsf{msg}}^{n \to i})$ be the message from Q in Round 3.
   - If $\mathsf{Verify}(\mathsf{vkey}_n, \mathsf{msg}^{n \to i}, \sigma_{\mathsf{msg}}^{n \to i}) \neq 1$ (OR) $\mathsf{msg}^{n \to i}$ is not of the form $(\text{“}\mathsf{Party\ 1}\text{ ”} \| m_1 \| \ldots \| \text{“}\mathsf{Party\ n}\text{ ”} \| m_n)$, send $\perp$ to Q and end the round.
   - Output $\perp$ to Q and end the round if there exists $j \neq n$ such that:
     - $\mathsf{msg}^{j \to i} \neq \mathsf{msg}^{n \to i}$ (AND)
     - $\mathsf{Verify}(\mathsf{vkey}_n, \mathsf{msg}^{j \to i}, \sigma_{\mathsf{msg}}^{j \to i}) = 1$ (AND)
     - $\mathsf{msg}^{j \to i}$ is of the form $(\text{“}\mathsf{Party\ 1}\text{ ”} \| m_1, \ldots, \| \text{“}\mathsf{Party\ n}\text{ ”} \| m_n)$. 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 $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$.
   - Parse $\mathsf{msg}^{n \to i}$ as $(\text{“}\mathsf{Party\ 1}\text{ ”} \| m_1, \ldots, \| \text{“}\mathsf{Party\ n}\text{ ”} \| m_n)$.
   - For each $j \in [n]$, do the following:

[9] Let $\mathcal{S} = \{i | \mathsf{ct}_i = \perp\}$. 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 in Round 1, P_i received $(\llbracket x_j \rrbracket, \pi_j, \sigma_j)$ from P_j such that $\mathsf{Verify}(\mathsf{vkey}_j, (\llbracket x_j \rrbracket, \pi_j), \sigma_j) = 1$ and $\mathsf{NIZK.Verify}(\pi_j, \mathsf{st}_j) = 1$ for $\mathsf{st}_j \in \mathsf{L}_1$ where $\mathsf{st}_j = (\llbracket x_j \rrbracket, \mathsf{pk})$, set $\mathsf{bit}_j = 1$. Else, set $\mathsf{bit}_j = 0$.
- If $m_j = \perp$:
  - * If $\mathsf{bit}_j = 1$, send $\perp$ to Q and end the round.
  - * Else, set $\mathsf{ct}_j = \perp$.
- If $m_j = (\llbracket x_j \rrbracket^{i_1 \to n}, \pi_j^{i_1 \to n}, \sigma_j^{i_1 \to n})$:
  - * If $\mathsf{Verify}(\mathsf{vkey}_j, (\llbracket x_j \rrbracket^{i_1 \to n}, \pi_j^{i_1 \to n}), \sigma_j^{i_1 \to n}) = 1$ and $\mathsf{NIZK.Verify}(\mathsf{crs}, \pi_j^{i_1 \to n}, \mathsf{st}_j) = 1$ for $\mathsf{st}_j \in \mathsf{L}_1$ where $\mathsf{st}_j = (\llbracket x_j \rrbracket^{i_1 \to n}, \mathsf{pk})$, set $\mathsf{ct}_j = \llbracket x_j \rrbracket^{i_1 \to n}$.
  - * Else, send $\perp$ to Q and end the round.

– Compute $\llbracket y \rrbracket \leftarrow \mathsf{dTFHE.Eval}(\mathsf{pk}, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_n)$.
– Compute $\llbracket y : \mathsf{sk}_i \rrbracket \leftarrow \mathsf{dTFHE.PartialDec}(\mathsf{sk}_i, \llbracket y \rrbracket)$ and $\pi_i^{\mathsf{dec}} \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}, \mathsf{st}_i^{\mathsf{dec}}, \mathsf{wit}_i^{\mathsf{dec}})$ for $\mathsf{st}_i^{\mathsf{dec}} \in \mathsf{L}_2$ where $\mathsf{st}_i^{\mathsf{dec}} = (\llbracket y : \mathsf{sk}_i \rrbracket, \llbracket y \rrbracket, \mathsf{pk}_i, i)$ and $\mathsf{wit}_i^{\mathsf{dec}} = (\mathsf{sk}_i, r_i)$.
– Send $(\llbracket y : \mathsf{sk}_i \rrbracket, \pi_i^{\mathsf{dec}})$ 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 $\mathsf{NIZK.Verify}(\mathsf{crs}, \pi_i^{\mathsf{dec}}, \mathsf{st}_i^{\mathsf{dec}}) \neq 1$ for $\mathsf{st}_i^{\mathsf{dec}} \in \mathsf{L}_2$ where $\mathsf{st}_i^{\mathsf{dec}} = (\llbracket y : \mathsf{sk}_i \rrbracket, \llbracket y \rrbracket, \mathsf{pk}_i, i)$, discard $\llbracket y : \mathsf{sk}_i \rrbracket$.
   – Output $y \leftarrow \mathsf{dTFHE.Combine}(\mathsf{pk}, \{\llbracket y : \mathsf{sk}_i \rrbracket\}_{i \in S})$ where S contains the set of non-discarded values from the previous step.

5.3 $(t+2)$ Round Protocol

We now describe how to transform the two-round protocol (say Π) of [28] into a $(t+2)$-round protocol Π' for solitary MPC with god. Recall that protocol Π (that assumes a PKI setup) achieves god for standard MPC and involves communication only via broadcast channels in both rounds. We propose the following changes to Π. First, we employ a $(t+1)$-round protocol over pairwise-private channels that realizes the broadcast functionality [17] to execute Round 1 of Π. Next, the messages communicated via broadcast in Round 2 of Π are instead sent privately only to Q (as only Q is supposed to obtain output) in Round $(t+2)$ of Π'. This completes the high-level description of Π' whose security follows directly from security of Π. This approach achieves better round complexity than our general five-round construction (Sect. 5.2) when $t \leq 2$.

Acknowledgments. We would like to thank the anonymous reviewers for their helpful and constructive comments on the manuscript. P. Miao is supported in part by the NSF CNS Award 2247352, a DPI Science Team Seed Grant, a Meta Award, and a DSI Seed Grant. All the authors did part of the work while at Visa Research.

References

1. Abraham, I., Devadas, S., Dolev, D., Nayak, K., Ren, L.: Synchronous byzantine agreement with expected O(1) rounds, expected o(n^2) communication, and optimal resilience. In: FC (2019)
2. Alon, B., Cohen, R., Omri, E., Suad, T.: On the power of an honest majority in three-party computation without broadcast. In: TCC (2020)
3. Ananth, P., Choudhuri, A.R., Goel, A., Jain, A.: Round-optimal secure multiparty computation with honest majority. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10992, pp. 395–424. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_14
4. Asharov, G., Beimel, A., Makriyannis, N., Omri, E.: Complete characterization of fairness in secure two-party computation of Boolean functions. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9014, pp. 199–228. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46494-6_10
5. Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., Wichs, D.: Multiparty computation with low communication, computation and interaction via threshold FHE. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 483–501. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_29
6. Badrinarayanan, S., Jain, A., Manohar, N., Sahai, A.: Threshold multi-key FHE and applications to round-optimal MPC. In: ASIACRYPT (2020)
7. Badrinarayanan, S., Miao, P., Mukherjee, P., Ravi, D.: On the round complexity of fully secure solitary mpc with honest majority. Cryptology ePrint Archive, Paper 2021/241 (2021). https://eprint.iacr.org/2021/241
8. Bell, J.H., Bonawitz, K.A., Gascón, A., Lepoint, T., Raykova, M.: Secure single-server aggregation with (poly)logarithmic overhead. In: CCS, pp. 1253–1269. ACM (2020)
9. Bonawitz, K., et al. Practical secure aggregation for privacy-preserving machine learning. In: CCS (2017)
10. Boneh, D., Gennaro, R., Goldfeder, S., Jain, A., Kim, S., Rasmussen, P.M.R., Sahai, A.: Threshold cryptosystems from threshold fully homomorphic encryption. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 565–596. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_19
11. Canetti, R., et al.: Fiat-shamir: from practice to theory. In: STOC (2019)
12. Chor, B., Merritt, M., Shmoys, D.B.: Simple constant-time consensus protocols in realistic failure models. J. ACM (JACM) **36**(3), 591–614 (1989)
13. Cleve, R.: Limits on the security of coin flips when half the processors are faulty (extended abstract). In: STOC (1986)
14. Cohen, R., Garay, J., Zikas, V.: Broadcast-optimal two-round MPC. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12106, pp. 828–858. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45724-2_28
15. Damgård, I., Magri, B., Ravi, D., Siniscalchi, L., Yakoubov, S.: Broadcast-optimal two round MPC with an honest majority. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12826, pp. 155–184. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84245-1_6
16. Damgård, I., Ravi, D., Siniscalchi, L., Yakoubov, S.: Minimizing setup in broadcast-optimal two round MPC. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023. LNCS, vol. 14005, pp. 129–158. Springer, Heidelberg (2023). https://doi.org/10.1007/978-3-031-30617-4_5

17. Dolev, D., Strong, H.R.: Authenticated algorithms for Byzantine agreement. SIAM J. Comput. **12**(4), 656–666 (1983)
18. Feldman, P., Micali, S.: An optimal probabilistic algorithm for synchronous Byzantine agreement. In: Ausiello, G., Dezani-Ciancaglini, M., Della Rocca, S.R. (eds.) ICALP 1989. LNCS, vol. 372, pp. 341–378. Springer, Heidelberg (1989). https://doi.org/10.1007/BFb0035770
19. Fischer, M.J., Lynch, N.A.: A lower bound for the time to assure interactive consistency. Inf. Process. Lett. **14**(4), 183–186 (1982)
20. Fitzi, M., Garay, J.A.: Efficient player-optimal protocols for strong and differential consensus. In: Borowsky, E., Rajsbaum, S. (eds.) 22nd ACM PODC, pp. 211–220. ACM (2003)
21. Fitzi, M., Garay, J.A., Maurer, U., Ostrovsky, R.: Minimal complete primitives for secure multi-party computation. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 80–100. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_5
22. Garg, S., Goel, A., Jain, A.: The broadcast message complexity of secure multiparty computation. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11921, pp. 426–455. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-34578-5_16
23. Gennaro, R., Ishai, Y., Kushilevitz, E., Rabin, T.: The round complexity of verifiable secret sharing and secure multicast. In: STOC (2001)
24. Gennaro, R., Ishai, Y., Kushilevitz, E., Rabin, T.: On 2-round secure multiparty computation. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 178–193. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45708-9_12
25. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC (1987)
26. Goldwasser, S., Lindell, Y.: Secure multi-party computation without agreement. J. Cryptol. **18**, 247–287 (2005)
27. Gordon, S.D., Hazay, C., Katz, J., Lindell, Y.: Complete fairness in secure two-party computation. J. ACM **58**(6), 24:1–24:37 (2011)
28. Dov Gordon, S., Liu, F.-H., Shi, E.: Constant-round MPC with fairness and guarantee of output delivery. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 63–82. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48000-7_4
29. Halevi, S., Ishai, Y., Kushilevitz, E., Makriyannis, N., Rabin, T.: On fully secure MPC with solitary output. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019. LNCS, vol. 11891, pp. 312–340. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36030-6_13
30. Halevi, S., Lindell, Y., Pinkas, B.: Secure computation on the web: computing without simultaneous interaction. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 132–150. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_8
31. Karlin, A., Yao, A.: Probabilistic lower bounds for byzantine agreement. Unpublished document (1986)
32. Katz, J., Koo, C.Y.: On expected constant-round protocols for byzantine agreement. J. Comput. Syst. Sci. **75**(2), 91–112 (2009)
33. Lamport, L., Shostak, R.E., Pease, M.C.: The byzantine generals problem. ACM Trans. Program. Lang. Syst. (1982)
34. Lynch, N.A.: Distributed Algorithms. Morgan Kaufmann, Burlington (1996)
35. Mohassel, A., Zhang, Y.: Secureml: a system for scalable privacy-preserving machine learning. In: IEEE S & P (2017)

36. Patra, A., Ravi, D.: On the exact round complexity of secure three-party computation. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10992, pp. 425–458. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_15
37. Pease, M., Shostak, R., Lamport, L.: Reaching agreement in the presence of faults. J. ACM **27**(2), 228–234 (1980)
38. Peikert, C., Shiehian, S.: Noninteractive zero knowledge for NP from (plain) learning with errors. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11692, pp. 89–114. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26948-7_4
39. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: FOCS (1986)