Two Round Information-Theoretic MPC with Malicious Security

Prabhanjan Ananth[1], Arka Rai Choudhuri[2], Aarushi Goel[2], and Abhishek Jain[2]

[1] Massachusetts Institute of Technology, USA,
prabhanjan@csail.mit.edu
[2] Johns Hopkins University, USA,
{achoud,aarushig,abhishek}@cs.jhu.edu

Abstract. We provide the first constructions of *two round* information-theoretic (IT) secure multiparty computation (MPC) protocols in the plain model that tolerate any $t < n/2$ malicious corruptions. Our protocols satisfy the strongest achievable standard notions of security in two rounds in different communication models.

Previously, IT-MPC protocols in the plain model either required a larger number of rounds, or a smaller minority of corruptions.

1 Introduction

The ability to securely compute on private datasets of individuals has wide applications of tremendous benefits to society. The notion of secure multiparty computation (MPC) [37,26,9,14] provides a solution to the problem of computing on private data by allowing a group of mutually distrusting parties to jointly evaluate any function over their private inputs in a manner that reveals nothing beyond the output of the function.

Information-Theoretic MPC. Over the years, a large body of works have investigated the design of MPC protocols against computationally bounded as well as computationally unbounded adversaries. In this work, we focus on the latter, namely, MPC with *information-theoretic* (IT) security.

The seminal works of [9,14] established the first feasibility results for IT-MPC for general functionalities. These works also established that IT security for non-trivial functions is only possible when at most $t < n/2$ of the n parties are corrupted. In scenarios where honest majority is a viable assumption, IT-MPC protocols are extremely appealing over their computational counterparts. In particular, they are typically more efficient since they do not use any computational primitives. Furthermore, IT-MPC protocols achieve security in models such as concurrent composition [11] without relying on external trust [12].

Round Complexity. In this work, we investigate the minimal conditions necessary for IT-MPC in the plain model. We focus on *round complexity* – a well studied complexity measure in distributed protocol design. We consider the standard simultaneous-message model of communication for MPC where in any round, each party can send messages to other parties, depending upon the communication from the previous rounds. We consider security against *malicious* adversaries who may corrupt any subset of $t < n/2$ parties and use arbitrary strategy to decide their protocol messages.

It is well known that two rounds of communication are necessary for MPC [28]. We ask whether two rounds are *sufficient* for achieving IT security:

Does there exist two round IT-MPC for any $t < n/2$ corruptions?

The above question has remained open for the last three decades. In particular, while constant round IT-MPC protocols are known for any $t < n/2$ corruptions (e.g., [6,31]), the only known two round IT-MPC protocols are due to [31,34,29] who require two-thirds honest majority (as opposed to standard honest majority). We refer the reader to Section 1.3 for a comprehensive survey of prior work, and Section 1.1 for comparison with the recent works of [3,19,4].

1.1 Our Results

In this work, we resolve the above question in the affirmative.

I. Main Result. Our first result is a two-round IT-MPC protocol for NC^1 functions that tolerates any $t < n/2$ corruptions. In the case of *malicious* adversaries, our protocol achieves statistical security with abort – the standard notion of security (c.f. [25]) where an adversary may prevent the honest parties from learning the output by aborting the computation. In the setting of two rounds, this is known to be the strongest achievable standard notion of security [24].

In the case of *semi-honest* adversaries, our protocol achieves perfect security.

Theorem 1. *There exists a two round MPC protocol for NC^1 functions that achieves:*

- *Statistical security with abort against $t < n/2$ malicious corruptions.*
- *Perfect security against $t < n/2$ semi-honest corruptions.*

II. Protocols over P2P Channels. Our protocol in Theorem 1 necessarily uses both broadcast and private point-to-point (P2P) channels for achieving security against malicious adversaries.[3] We next investigate whether it is possible to construct two round IT-MPC against malicious adversaries by using *only* P2P channels.[4]

Our second result is a two round IT-MPC protocol over P2P channels that achieves statistical *security with selective abort* against any $t < n/2$ malicious corruptions. This notion [27] is a weakening of the standard notion of security of

[3] In the case of semi-honest adversaries, broadcasts can be trivially emulated over P2P channels without any increase in round complexity.

[4] Note that the complementary goal of IT-MPC over only broadcast channels is known to be impossible.

(unanimous) abort in that it allows the adversary to separately decide for each honest party whether it will receive the correct output or $\perp$. Achieving security with abort in two rounds over P2P channels is known to be impossible in general [18,35]. This establishes security with selective abort as the strongest achievable standard notion of security in two rounds.

Theorem 2. *There exists a two round MPC protocol over P2P channels for NC^1 functions that achieves statistical security with selective abort against $t < n/2$ malicious corruptions.*

Put together, Theorems 1 and 2 fully resolve the round complexity of maliciously secure IT-MPC (for NC^1 functions).

Comparison with [3,19,4]. Recently, Applebaum et al. [3] constructed two round perfectly secure MPC for NC^1 against any $t < n/2$ semi-honest corruptions. Garg et al. [19] achieve a similar result; however, the communication complexity of their protocols grows super-polynomially with the number of parties. Neither of these works consider security against malicious adversaries, which is the main focus of our work. A recent independent and concurrent work of Applebaum et al. [4] also considers the case of malicious adversaries. Similar to our work, they also construct a two-round statistically secure protocol for NC^1 functionalities that achieves security with selective abort. However, they do not achieve our main result, namely a two-round information-theoretic protocol for security with (unanimous) abort.

1.2 Technical Overview

We first focus on achieving two-round IT-secure MPC in the presence of both broadcast and point to point communication channels.

Recent works on two-round secure MPC [21,10,22] follow a common blueprint of squishing an arbitrary round secure protocol, referred to as *inner* protocol, into a two round secure protocol, referred to as *outer* protocol using garbled circuits. Roughly speaking, every party in the outer protocol computes t garbled circuits, one for every round of the inner protocol. The job of the j^{th} garbled circuit computed by the i^{th} party is to emulate the computation of the next message function of the i^{th} party in the j^{th} round. Every party sends the generated t garbled circuits to the other parties.

The main challenge here is to ensure that the garbled circuits can talk to each other the same way the parties in the inner protocol talk to each other. The tools used to address this challenge differs from one work to another: [21] use bilinear maps, [22] use two-round oblivious transfer, [10,20] use two-round oblivious oblivious transfer and additionally garbled circuits and finally, [1,3,19] use information-theoretic MPC protocols. Of particular interest to us is the work of Ananth et al. [1] who show how to achieve maliciously secure two-round secure MPC in the honest majority setting for polynomial-sized circuits assuming only one-way functions.

Background on [1]. They propose the following template: The first step is to construct *helper* protocols that enable communication between garbled circuits in the outer protocol. The helper protocols they consider are delayed-function two-round MPC protocols, handling malicious adversaries, for two functionalities defined below. In a delayed-function two-round MPC protocol, the functionality is only available to the parties after the first round.

- The first functionality, parameterized by a bit v, is defined as follows: it takes as input r_1 from the first party, r_2 from the second party and outputs $r_1 \oplus r_2 \oplus v$.
- The second functionality, parameterized by two bits (v_1, v_2), is defined as follows: it takes as input a string K from the first party (interpreted as an input wire label of a garbled circuit), three bits (r_1, r_2, r_3) from the second party and outputs $K_{r_3 \oplus NAND(v_1 \oplus r_1, v_2 \oplus v_3)}$.

Observe that both these functionalities can be represented by quadratic polynomials over $\mathbb{F}_2$ and there exist two-round protocols for quadratic polynomials in the literature (see [34]). While these protocols do not achieve full-fledged malicious security, they achieve a weaker property termed as *privacy with knowledge of outputs* and [1] show how this weaker property is sufficient for their goal.

The next step is to transform the inner interactive protocol into an outer two-round protocol using the helper protocols. Since the helper protocols can only compute restricted functionalities, they impose a restriction on the "structure" of the inner protocol. In particular, every round of the inner interactive protocol is forced to only perform a single NAND computation. The term conforming protocols (originally coined by [22]) was used to described such interactive protocols.

Informally, a conforming protocol proceeds in a sequence of rounds. In every round, a party, termed as "receiver", obtains a global state from another party, termed as "sender", that encodes information about the current states of all the parties. Every party possesses a decryption key that lets it decode only a certain section of the global state. Once the party decodes the appropriate information, it then performs some local computation and then re-encodes the result and the resulting updated global state will be broadcasted to the rest of the parties, termed "listeners". Thus in every round, there is a sender, receiver and the rest of the parties are listeners.

At first, it might seem unclear as to why conforming protocols should exist at all. Luckily, an arbitrary round information-theoretically secure protocol can be transformed into a conforming protocol. However, the transformation demonstrated by [1] blows up the round complexity of the conforming protocol. In particular, *even if the original protocol had a constant number of rounds, the corresponding conforming protocol will now have round complexity proportional to the size of the circuit being securely computed.* Nevertheless, their transformation from a conforming protocol into the two-round outer protocol for polynomial-sized circuits is unaffected by the round complexity of the underlying conforming protocol.

Limitations on extending [1] to IT setting. To construct maliciously secure information-theoretically secure MPC protocols for NC^1 circuits, a natural direction to explore is to adapt the construction of [1] to the information-theoretic setting. The only part in the construction where one-way functions are used is in the generation of garbled circuits. If we restrict to NC^1 circuits, we could hope to use garbling schemes with perfect security [32]. These garbling schemes have the property that the size of the wire labels for the input wires grows exponentially in the depth of the circuit being garbled and linearly in the size of the garbled circuit.

This results in a fundamental issue in using information-theoretic garbling schemes to replace the garbled circuits based on one-way functions in [1]: as part of the outer protocol, every party sends a sequence of garbled circuits, where every garbled circuit encodes wire labels for the next garbled circuit. Recall that every garbled circuit emulates the next message function in a round and it needs to encode the wire labels for the next garbled circuit to enable transferring information from one round to the next. Once we use information-theoretically secure garbling schemes, the communication complexity now blows up exponentially in the length of the chain of garbled circuits. Since the length of the chain is the round complexity of the underlying conforming protocol, this results in *exponential communication complexity* even for NC^1 functionalities.

Our Approach. As a first step towards achieving our goal, we consider conforming protocols that do not restrict every round in the outer protocol to be just a single NAND computation. More generally, we allow the next message in every round of the conforming protocol to be a polynomial-size NC^1 circuit. We term this class of protocols to be *generalized* conforming protocols. On the one hand, the advantage of considering generalized conforming protocols is that we can construct this in constant number of rounds for NC^1 which makes it suitable to use it towards constructing a two-round protocol in the information-theoretic setting. On the other hand, the helper protocols designed in [1] are no longer compatible with our notion of generalized conforming protocols; recall that since the helper protocols in [1] were associated with quadratic polynomials, they imposed the requirement that every round in the conforming protocol is a single NAND computation.

To address this issue, we design *new* helper protocols that are "compatible" with generalized conforming protocols. Specifically, we require that the helper protocols are associated with functionalities computable in NC^1. By carefully examining the interiors of [1], it can be observed that it suffices to construct helper protocols for *three*-input functionalities computable in NC^1; these are the functionalities where only three parties have inputs. Informally, the three parties correspond to a sender party that sends a message in a round, a receiver party that receives a message in a round and finally, a listener party that listens to the communication from the sender to the receiver. Even though there are multiple listeners in every round in the conforming protocol, it suffices to design helper protocols for every listener separately. In the helper protocol, the inputs of the sender and the receiver are their private states[5] and the listener's input would be the wire labels for its garbled circuits. Note, however, that *the functionality associated with the helper protocol is as complex as the next message function of the conforming protocol.*

As such, it is unclear how to construct helper protocols even for three-input functionalities; in fact, if we had a two-round secure protocol for the three-input functionality that outputs the product of its inputs, then it could be bootstrapped to achieve two-round secure protocols for arbitrary functionalities via randomized encodings [31]. In light of this, the problem of constructing two-round secure protocols for three-input functionalities seems as hard as constructing two-round secure protocols for all functionalities computable in NC^1.

We resolve this dilemma in two main steps:

- We first focus on a weaker goal: constructing two-round information theoretically secure protocols for *two-input* (as opposed to three-input) functionalities.
- We then go back to our definition of generalized conforming protocols and impose additional structure on generalized conforming protocols – without blowing up their round complexity – to make them compatible with helper protocols for two-input functionalities.

We start by defining and constructing helper protocols for two-input functionalities.

Helper Protocols for Two-Input Functionalities. A two-input multiparty functionality, as the name suggests, is a functionality where only the first two parties get inputs while the rest of the parties are input-less. We consider two-input functionalities of the following form: these functionalities $\mathcal{U}$ are parameterized by two NC^1 functions f, G such that $\mathcal{U}(x_1, x_2, \perp, \cdots, \perp) = G(x_1, f(x_2))$. At first sight, this representation may seem unnecessary since one can rewrite $\mathcal{U}$ as another NC^1 function G' such that $\mathcal{U}(x_1, x_2, \perp, \cdots, \perp) = G'(x_1, x_2)$. However, the functions G and f we use to express $\mathcal{U}$ makes a difference when we state the security guarantees. Moreover, we require that the resulting helper protocol satisfies *delayed-function* property, meaning that the functionalities is only available to the parties after the first round.

Informally, we require the following asymmetric security guarantees:

- If the first party is honest then no information about its input x_1 should be leaked beyond $G(x_1, y^*)$. Ideally, we would require y^* to be the output of f on some input x_2^*. Here, we relax the security requirement to allow y^* to not even belong in the range of f.
- If the second party is honest then no information about its input x_2 should be leaked beyond $f(x_2)$. In particular, we allow the adversary to learn the

[5] Since the listener listens to the conversation, the receiver and the sender would share a secret string in order to emulate communication over *private channels (which are necessary for information-theoretic security).* This is the reason why the receiver should also input its private state.

value $f(x_2)$ during the execution of the protocol. In addition, we only require that the simulator extracts the implicit input (interpreted as $f(x_2)$) and not x_2 itself.

Both the security requirements are non-standard and indeed, its should not be clear in what context these two security properties would be useful. To answer this, lets recall the structure of the conforming protocol: in every round, every party receives a global state, decodes a portion of the global state, computes on it and re-encodes the result. Looking ahead, when the conforming protocol is used alongside the helper protocols, the function f would have the global state hardwired inside its code; it takes as input private state of the party, represented by x_2, performs computation and then re-encodes the result. So the output $f(x_2)$ denotes the resulting global state.

Let us revisit the security requirements stated above. Allowing for y^* to not be in the range of f reduces to allowing for the second party to be malicious in the conforming protocol. We handle this by designing conforming protocols already secure against malicious parties. Regarding the second security requirement, revealing the value $f(x_2)$ reduces to the party revealing the updated global state. Since a party anyways has to broadcast the entire global state in the conforming protocol, its perfectly safe to reveal $f(x_2)$.

We now give a glimpse of our construction of two-round protocol for two-input functionalities. Our construction is heavily inspired by the techniques introduced in the work of Benhamouda and Lin [10].

- In the first round, the second party holding the input x_2, sends a garbling GC_2 of a universal circuit with x_2 hardwired inside it. The first party, holding the input x_1, receives GC_2 and computes another garbling GC_1 of a circuit, with x_1 hardwired inside it, that is defined as follows: it takes as input, wire labels of GC_2 with respect to input f, evaluates GC_2 using these input wire labels to obtain $f(x_2)$ and finally outputs $G(x_1, f(x_2))$.
- Simultaneously, all the parties execute a secure MPC protocol for quadratic polynomials, that takes as input wire labels of GC_2 from the second party, input wire labels of GC_1 from the first party and finally, computes GC_1 input wire labels associated with the input which is in turn defined to be the GC_2 input wire labels associated with f.

At the end of the second round, every party evaluates GC_1 to obtain $G(x_1, f(x_2))$.

We briefly describe the simulation strategy for arguing security of the above construction. If the second party is corrupted then the simulator extracts all the wire labels of GC_2 and then evaluates GC_2 using the wire labels of f to obtain the value y^*. The simulator then sends y^* to the ideal functionality, which responds back with $G(x_1, y^*)$. The simulator cannot verify that the second party indeed sent a valid garbling of the universal circuit. However, this still satisfies our security definition since the simulator is not required to extract x_2 but only the value y^*.

The case when the first party is corrupted can similarly be argued by designing a simulator that first extracts all the wire labels of GC_1 and then simulates GC_2 using the value $f(x_2)$.

CLC property of Generalized Conforming Protocols. As explained earlier, helper protocols for two-input functionalities is as such incompatible with our current definition of generalized conforming protocols. Recall that the reason for incompatibility was that in every round of the generalized conforming protocol there were three parties participating. To remedy this situation, we introduce a new structural property for generalized conforming protocols, that we refer to as *copy-local-copy* (CLC) property. Specifically, we require that a party in every round, behaves as follows:

- Copy operation: first, every party copies the information transferred on the communication channels onto its own private state.
- Local computation: then it performs computation on its own local state.
- Copy operation: finally, it copies the result obtained onto the communication channel.

The CLC property effectively "breaks down" each three-input computation required in the earlier notion of generalized conforming protocol into three different operations. Now, given a generalized conforming protocol that satisfies the CLC property, it suffices to devise helper protocols for the above three operations.

The helper protocols for the first copy operation, and also the third copy operation, are associated with three parties: speaker, receiver and the listener. However, since the copy operation is a simple function, we observe that it suffices to use helper protocols for quadratic polynomials to implement this. The helper protocol for the local computation, however, is only associated with two parties: the party performing the local computation and the listener. Now, we use the delayed-function secure protocol for two-input functionalities constructed earlier to realize helper protocols associated with the local computation operation.

Since we divide every round of the protocol into three parts, a party sends three garbled circuits for every round of the conforming protocol, instead of just one.

Summary. We now summarize the main steps in the construction of maliciously secure information-theoretically secure multiparty protocols for NC^1 functionalities.

- First, we consider delayed-function two round secure MPC protocols for quadratic polynomials in Section 3.1.
- Then we define the notion of delayed-function two round secure MPC protocols for two-input NC^1 functionalities in Section 3.2. We define the security requirements in Section 3.2. This is followed by a construction of this notion in Section 3.2.
- In Section 4, we define the notion of generalized conforming protocols. We state the CLC property in Definition 7.
- Finally, we present the main construction in Section 5.

Protocol over P2P Channels. Next, we focus on designing a two-round protocol over P2P channels that achieves security with selective abort against malicious adversaries. Recall that in security with selective abort, the adversary can selectively decide which of the honest parties can receive the output while the rest of them abort. However, the adversary cannot force an “invalid” output on any of the honest parties.

To achieve our goal, we start with a two-round protocol Π_{in} over broadcast and P2P channels satisfying security with (unanimous) abort. A naive attempt would be as follows: start with Π_{in} and whenever a party has to send a broadcast message, he instead sends this message over P2P channels to all the other parties. Note that the resulting protocol is over P2P channels. However, this doesn’t work: there is no mechanism in place to ensure that a malicious party indeed sends the *same* message, originally a broadcast message in Π_{in}, to all the other parties over P2P channels. The protocol Π_{in} might not be resilient to such attacks which would result in our resulting protocol to be insecure.

We introduce mechanisms to prevent this attack. Towards this, our idea is to require each party to send a garbled circuit of (a slightly modified version of) their second round next message function in Π_{in} in the second round of the P2P channel protocol. This (modified) next message function has the party’s input and randomness, and the private channel messages that the party received in the first round of Π_{in} hard-wired inside its description. It additionally takes the first round broadcast channel messages of Π_{in} as input. To enable other parties to evaluate this garbled circuit, we require each party to send additive secret shares of all the labels for its garbled circuit over private channels (in particular, each party only receives one of the shares for each label) in the first round itself. In the second round, each party simply reveals the appropriate shares for each garbled circuit based on the messages received in the first round. If the adversary does not send the same set of broadcast messages to all parties, each party will end up revealing shares corresponding to a different label. In this case, we rely on the security of garbled circuits to ensure that nobody (including the adversary) is able to evaluate any of the honest party garbled circuits.

However, there are some subtle issues that crop when implementing this approach:

- Since we want the resulting protocol to satisfy information-theoretic security, we require the next-message function of Π_{in} to be computable in NC^1.
- The transformation sketched above does not handle the case when Π_{in} sends messages over private channels in the second round.

Fortunately, the information-theoretically secure MPC protocol over broadcast and P2P channels that we constructed earlier satisfies both the above properties and thus can be used to instantiate Π_{in} in the above approach. This gives us a P2P channel two-round MPC protocol that achieves security with selective abort against malicious adversaries. We present the construction of this protocol in section 6.

1.3 Related Work

Since the initial feasibility results [37,26,9,14], a long sequence of works have investigated the round complexity of MPC. Here, we focus on protocols in the honest majority setting, and refer the reader to [5] for a survey of related works in the dishonest majority setting.

Information-Theoretic MPC. The seminal works of [9,14] provided the first constructions of polynomial-round IT-MPC protocols for general functionalities. These results were further improved upon in [7,36,13] w.r.t. malicious corruption threshold.

Bar-Ilan and Beaver [6] initiated the study of constant-round IT-MPC protocols. Subsequently, further improvements were obtained by [17,30,15]. The work of [31] provided the first constructions of two and three round IT-MPC protocols against $t < n/3$ and $t < n/2$, respectively, semi-honest corruptions. In the three round setting, their work was extended to handle a constant fraction of malicious adversaries by [23]. [32] constructed constant round perfectly secure protocols, improving upon the work of [6]. More recently, two round IT-MPC protocols that achieve security with selective abort against $t < n/3$ malicious corruptions were constructed by [34] and [29]. In fact, [34] and [29], put together, also achieve the stronger notion of security with guaranteed output delivery for the specific case of $n \geqslant 4$ parties and $t = 1$ corruptions which is not covered by the impossibility results of [18,35]. All of these positive results are for NC^1 functions; [33] established the difficulty of constructing constant-round IT-MPC protocols for general functionalities.

We also highlight the work of [27] who provided a general compiler to transform protocols over broadcast channels that achieve security with abort into protocols over P2P channels that achieve security with selective abort. Their transformation is unconditional, and increases the round-complexity by a multiplicative factor of three.

Computationally secure MPC. The study of constant-round computationally secure MPC protocols in the honest majority setting was initiated by Beaver et al. [8] who constructed such protocols for general functionalities based on one-way functions. Damgård and Ishai [16] provided improved constructions based on only black-box use of one-way functions.

Two round protocols for general functionalities against $t < n/3$ malicious corruptions were constructed by [34] and [29] based on one-way functions. Very recently, Ananth et al. [1] constructed two round protocols for general functionalities that achieve security with abort against any $t < n/2$ malicious corruptions based on black-box use of one-way functions. Applebaum et al. [3] and Garg et al. [19] also achieve similar results, albeit only against semi-honest adversaries.

2 Preliminaries

We denote the statistical security parameter by $\mathbf{k}$. We use the standard notion of security with abort for multi-party computation against malicious adversaries.

For our second result over P2P channels, we consider a weaker notion of security, call security with selective abort. In security with selective abort, the adversary can selective cause some honest parties to output $\perp$. Note that this is slightly different from the standard notion of security with abort, where the adversary can only either allow all honest parties to learn the output or cause all the honest parties to output $\perp$.

We also consider an even weaker notion of security called privacy with knowledge of outputs where the privacy of honest parties' inputs is ensured but the correctness of output for the honest parties is not guaranteed. We also use statistically secure garbled circuits [37] in our protocols.

3 Helper Two-round Secure Protocols

We consider two types of helper protocols towards achieving our main goal:

- First, we consider a two-round secure multiparty computation protocol for NC^1 two-input functionalities; that is, only two of the parties have inputs. We consider this notion in the delayed-function setting.
- Next, we consider a two-round secure multiparty computation protocol for quadratic polynomials, also in the delayed function setting.

3.1 Delayed-Function Two-Round Secure MPC for Quadratic Polynomials

A delayed-function two-round secure MPC protocol is a special case of maliciously secure two-round secure MPC where the functionality is available to the parties only after the first round. One of the helper tools we use is a two-round secure MPC protocol for quadratic polynomials in the delayed function setting. Such a result was already shown by Ishai et al. [34]. Formally, they prove the following lemma.

Lemma 1 ([34]). *Let $n > 0$ and $\ell_{out} > 0$. Consider a n-party functionality $G : \{0,1\} \times \cdots \times \{0,1\} \to \mathcal{Y}^{\ell_{out}}$, where $\mathcal{Y} = \{(0,\ldots,0),(1,\ldots,1)\}$, and every output bit of G is computable by an n-variate quadratic polynomial over $\mathbb{F}_2$. There is a delayed-function two-round MPC protocol for G satisfying perfect privacy with knowledge of outputs property in the honest majority setting. Moreover, the next message of this protocol can be represented by a $O(\log(n))$-depth $(\ell_{out} \cdot n)^c$-sized circuit, for some constant c.*

Remark. The protocol of [34] only guarantees a weaker variant of privacy with knowledge of outputs where the adversary can force different honest parties to output different values. However if we use a broadcast channel in the second round, their protocol achieves a stronger variant of privacy with knowledge of outputs, where all honest parties learn the same output.

3.2 Delayed-Function Two-round Secure MPC

The other helper tool we require is a delayed-function secure MPC protocol for arbitrary functionalities, but where only two parties have inputs. In particular, we are interested in the class of functionalities $\{F_{G,f}\}$: each functionality $F_{G,f}$ is parameterized by two functions G, f; it takes as input $(x_1, x_2, \perp, \ldots, \perp)$ and outputs $F_{G,f}(x_1, x_2, \perp, \ldots, \perp) = G(x_1, f(x_2))$. That is, party P_1 gets as input x_1 and party P_2 gets as input x_2. If the functionality $F_{G,f}$ were to be available to the parties before the protocol begins then securely computing $F_{G,f}$ would reduce to securely computing G since P_2 can pre-compute $f(x_2)$ and then run the secure protocol for G. However, we consider delayed-function setting and so this would not work.

In terms of security, we require the following informal guarantees.

- Security against P_2: unlike the standard simulation-based paradigm, in the ideal world, the honest parties and the simulator only have oracle access to G. In particular, the simulator only has to extract the value y (termed as *true* input of P_2), interpreted as the output of f on some input x_2 (also called *implicit* input of P_2), from the adversary.
- Security against P_1: we require that the implicit input x_2 of P_2 is hidden from P_1. However, we don't enforce that the output $f(x_2)$ is hidden from P_1. Moreover, we require the input privacy of P_2 to hold even if P_1's behaviour deviates from the protocol.

In particular, we require different security guarantees depending on which party the adversary corrupts.

Two-Input Multiparty Functionalities We consider delayed-function two-round secure MPC protocols, where the parties determine the functionality (to be computed on their private inputs) only after the first round. This notion is referred as delayed-function secure MPC protocols in the literature. We describe the class of functionalities that we are interested in. Later, we define the security properties associated with delayed-function secure MPC protocols for this class of functionalities.

Two-Input n-Party Functionalities. A two-input n-party functionality is an n-party functionality where only two parties receive inputs from the environment.

Definition 1 (Two-Input n-Party Functionality). *Let $n, \ell_1, \ell_2, \ell' > 0$. We define an n-party functionality G to be a two-input functionality if its of the following form: it takes as input from the domain $\{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2} \times \perp \times \cdots \times \perp$ and outputs a value in $\{(y, \ldots, y)\}_{y \in \{0,1\}^{\ell'}}$.*

We are interested in a sub-class of two-input functionalities that we refer to as specialized two-input n-party functionalities. Every functionality in this class, on input $(x_1, x_2, \perp, \ldots, \perp)$, first performs pre-processing on one of the inputs, say x_2, and then performs computation on the preprocessed result and x_1. The reason why we differentiate between pre-processing and post-processing becomes clear later on, when we define security against adversarial P_2.

Definition 2 (Specialized Two-Input n-Party Functionality). *Let $n, \ell_1, \ell_2, \ell' > 0$. We define an n-party functionality mapping $\{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2} \times \perp \cdots \times \perp$ to $\{(y, \ldots, y)\}_{y \in \{0,1\}^{\ell'}}$ (parameterized by a functions G and f) to be a specialized two-input functionality if its of the following form: it takes as input $(x_1, x_2, \perp, \ldots, \perp)$ and outputs $G(x_1, f(x_2))$.*

Security Let $P_1, \ldots, P_n$ be the parties participating in the delayed-function secure MPC protocol. We consider three cases and define separate security properties for each of these three cases: (i) P_1 is in the corrupted set while P_2 is not, (ii) P_2 is in the corrupted set while P_1 is not and, (iii) neither P_1 nor P_2 is in the corrupted set. Note that we don't consider the case when P_1 and P_2 are both in the corrupted set because P_1 and P_2 are the only parties receiving inputs in the protocol. We note that in all the three cases we are required to handle adversaries that deviate from the behavior of the protocol.

We define the following set systems.

- $\mathcal{S}_1 = \big\{T \subseteq \{P_1, \ldots, P_n\} : |T| < \lfloor \frac{n}{2} \rfloor, P_1 \in T, P_2 \notin T\big\}$
- $\mathcal{S}_2 = \big\{T \subseteq \{P_1, \ldots, P_n\} : |T| < \lfloor \frac{n}{2} \rfloor, P_1 \notin T, P_2 \in T\big\}$
- $\mathcal{S}_3 = \big\{T \subseteq \{P_1, \ldots, P_n\} : |T| < \lfloor \frac{n}{2} \rfloor, P_1 \notin T, P_2 \notin T\big\}$

We now handle the three cases below. Denote S to be the corrupted set of parties. Let x_1 and x_2 be the inputs of P_1 and P_2 respectively.

Case 1. $S \in \mathcal{S}_1$. To define the security property for this case, we consider two experiments Expt_0 and Expt_1. In Expt_0, the honest parties and the adversary execute the protocol (real world). The output of Expt_0 is the view of the adversary and the outputs of the honest parties.

In Expt_1, the corrupted set of parties execute the protocol with the rest of the parties, simulated by a PPT algorithm Sim. In the first round, the simulator does not get any input and after the first round, the simulator gets as input $f(x_2)$, where $F_{G,f}$ is the n-party functionality associated with the protocol. The output of Expt_1 is the view of the adversary and the output of the simulator.

We require that the output distributions of the experiments Expt_0 and Expt_1 are identically distributed.

Definition 3 (Security Against $\mathcal{S}_1$). *Consider a delayed-function n-party protocol Π for a class of specialized two-input n-party functionalities $\{F_{G,f}\}$ mapping $\{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2} \times \perp \cdots \times \perp$ to $\{(y, \ldots, y)\}_{y \in \{0,1\}^{\ell'}}$. We say that Π is secure against $\mathcal{S}_1$ if for every adversary corrupting a set of parties $S \in \mathcal{S}_1$, there exists a PPT simulator Sim such that the output distributions of Expt_0 and Expt_1 are identically distributed.*

Case 2. $S \in \mathcal{S}_2$**.** We handle this case using the real world-ideal world paradigm. In the real world, the corrupted parties and the honest parties execute the protocol. The output of the real world is the view of the adversary and the outputs of the honest parties. In the ideal world, the honest parties and the simulator have oracle access to the n-party functionality G [6]. The output of the ideal world are the outputs of the honest parties and the output of the simulator.

More formally, we can define the real world process $\mathsf{Real}^{\mathcal{A},F}$ and the ideal world process $\mathsf{Ideal}^{\mathsf{Sim},G}$– in particular, as in the definition of privacy with knowledge of outputs property, the simulator directs the trusted party to deliver outputs, of its choice, to the honest parties.

We define security of delayed-function secure MPC protocols against $\mathcal{S}_2$.

Definition 4 (Security Against $\mathcal{S}_2$). *Consider a delayed-function n-party protocol Π for a class of specialized two-input n-party functionalities $\{F_{G,f}\}$ mapping $\{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2} \times \perp \cdots \times \perp$ to $\{(y,\ldots,y)\}_{y \in \{0,1\}^{\ell'}}$. We say that Π is secure against $\mathcal{S}_2$ if for every adversary $\mathcal{A}$ corrupting a set of parties $S \in \mathcal{S}_2$, there exists a PPT simulator* Sim *such that the output distributions of* $\mathsf{Real}^{\mathcal{A},F}(x_1,\ldots,x_n)$ *and* $\mathsf{Ideal}^{\mathsf{Sim},G}(x_1,\ldots,x_n)$ *are identically distributed.*

Remark 1. Since the simulator only has access to the ideal functionality of G (and not F) in the ideal world, this means that the simulator is required to only extract the *implicit* input (and not the *true* input) of the adversary. In particular, if f is the identity function, then this security notion implies the standard simulation-based security.

Case 3. $S \in \mathcal{S}_3$**.** In this case, we require the protocol to satisfy privacy with knowledge of outputs property. Formally, we can analogously define the real world process $\mathsf{Real}^{\mathcal{A},F}$ and ideal world process $\mathsf{Ideal}^{\mathsf{Sim},F}$. We define the security property below.

Definition 5 (Security Against $\mathcal{S}_3$). *Consider a delayed-function n-party protocol Π for a class of specialized two-input n-party functionalities $\{F_{G,f}\}$ mapping $\{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2} \times \perp \cdots \times \perp$ to $\{(y,\ldots,y)\}_{y \in \{0,1\}^{\ell'}}$. We say that Π is secure against $\mathcal{S}_3$ if for every adversary $\mathcal{A}$ corrupting a set of parties $S \in \mathcal{S}_3$, there exists a PPT simulator* Sim *such that the output distributions of* $\mathsf{Real}^{\mathcal{A},F}(x_1,\ldots,x_n)$ *and* $\mathsf{Ideal}^{\mathsf{Sim},F}(x_1,\ldots,x_n)$ *are identically distributed.*

We are now ready to formally define a delayed-function secure MPC protocol for specialized two-input functionalities.

Definition 6. *Consider a delayed-function n-party protocol Π for a specialized two-input n-party functionality. We say that Π is secure if Π is secure against $\mathcal{S}_1$ (Definition 3), secure against $\mathcal{S}_2$ (Definition 4) and secure against $\mathcal{S}_3$ (Definition 5).*

[6] We emphasize that the parties have oracle access to G and not F.

Construction We prove the following lemma.

Lemma 2. *Let $n, \ell_1, \ell_2, \ell' > 0$. Consider a two-input n-party functionality $G : \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2} \times \perp \times \cdots \times \perp \to \{(y, \ldots, y)\}_{y \in \{0,1\}^{\ell'}}$ computable by a depth-d circuit of size s. There is a delayed-input two-round MPC protocol for a specialized two-input functionality G (Definition 2) satisfying perfect privacy with knowledge of outputs property in the honest majority setting. Moreover, the next message function of the every party in the protocol can be represented by a circuit of depth $O(d + \log(s))$ and size $s^c 2^{c \cdot (d + \log(s))}$, for some constant c.*

Proof. The main tools used in the construction are a perfectly secure garbling scheme and a secure MPC protocol for quadratic polynomials in the honest majority setting satisfying privacy with knowledge of outputs property (Lemma 1). We denote the garbling scheme by $(\mathsf{Gen}, \mathsf{Garb}, \mathsf{Eval})$. We denote the secure MPC protocol for quadratic polynomials by $\mathit{\Pi}_{\mathsf{Quad}}$.

We construct a delayed-function secure MPC protocol for a class of specialized two-input functionalities $\{F_{G,f}\}$, each functionality implementable by a circuit of size s and depth d. Our construction is heavily inspired by the techniques introduced in the work of Benhamouda and Lin [10]. Suppose P_1 has input x_1, P_2 has input x_2 and the rest of the parties don't receive any input. The protocol proceeds as follows: set the statistical security parameter, $\mathbf{k} = 1$.

Round 1.

- P_1 generates $\mathsf{Gen}(1^{\mathbf{k}}, 1^{L'}, 1^{d'})$ to obtain $(\mathsf{gk}_1, \mathbf{K}_I^1)$, where L' and d' are defined below. It also generates the first round messages of $\mathit{\Pi}_{\mathsf{Quad}}$. In $\mathit{\Pi}_{\mathsf{Quad}}$, its input is $\mathbf{K}_I^1$. It sends the first round messages of $\mathit{\Pi}_{\mathsf{Quad}}$ to other parties.

- P_2 generates $\mathsf{Gen}(1^{\mathbf{k}}, 1^{L''}, 1^{d''})$ to obtain $(\mathsf{gk}_2, \mathbf{K}_I^2)$, where L'' and d'' (defined in first of Round 2). It also generates the first round messages of $\mathit{\Pi}_{\mathsf{Quad}}$. It also generates a random string R (we define its length below). In $\mathit{\Pi}_{\mathsf{Quad}}$, its input is $(\mathbf{K}_I^2 \circ R)$. It generates $\mathsf{Garb}(\mathsf{gk}_2, U_{x_2})$ to obtain GC_2, where U_{x_2} is a universal circuit with x_2 hardwired in it, it takes as input a circuit of size s, depth d and outputs a single bit. Set $|R| = |\mathsf{GC}_2|$. Note that U_{x_2} can be implemented by a circuit of size $L'' = O(s)$ and depth $d'' = O(d)$. It sends $\mathsf{GC}_2 \oplus R$ along with the first round messages of $\mathit{\Pi}_{\mathsf{Quad}}$ to other parties.

- P_i, for $i \neq 1, i \neq 2$, generates the first round messages of $\mathit{\Pi}_{\mathsf{Quad}}$. It sends the first round messages to other parties.

Round 2. At the end of round 1, the parties receive the function f as input.

- P_1 generates the second round messages of $\mathit{\Pi}_{\mathsf{Quad}}$. The protocol $\mathit{\Pi}_{\mathsf{Quad}}$ is associated with a function that takes as input $(\mathbf{K}_I^1, \mathbf{K}_I^2, \perp, \ldots, \perp)$ and outputs $\mathbf{K}_I^1 \left[\mathbf{K}_I^2[f] \circ R\right]$[7]. We note that this function can be implemented by a system

[7] Recall that the notation $\mathbf{K}_I^1 \left[\mathbf{K}_I^2[f] \circ R\right]$ refers to the input wire labels for GC_1 corresponding to the input $(\mathbf{K}_I^2[f] \circ R)$. Moreover, $\mathbf{K}_I^2[f]$ refers to the input wire labels for GC_2 corresponding to the input f.

of quadratic polynomials over $\mathbb{F}_2$. It generates $\mathsf{Garb}(\mathsf{gk}_1, \widehat{G})$ to obtain GC_1, where $\widehat{G}$ (with $\mathsf{GC}_2 \oplus R$ hardwired) is defined as follows: it takes as input $(\mathbf{K}^2_I[f], R)$, computes $y \leftarrow \mathsf{Eval}(\mathsf{GC}_2, \mathbf{K}^2_I[f])$ and finally it outputs $G(x_1, y)$. $\widehat{G}$ can be implemented by a circuit of size $L' = O(s)$ and depth $d' = O(d)$. P_1 sends the second round messages of $\mathit{\Pi}_{\mathsf{Quad}}$ along with GC_1.

- P_2 generates the second round messages of $\mathit{\Pi}_{\mathsf{Quad}}$ and sends them to other parties.
- P_i, for $i \neq 1$ and $i \neq 2$, computes the second round messages of $\mathit{\Pi}_{\mathsf{Quad}}$ and sends them to other parties.

Reconstruction. All the parties compute the output of $\mathit{\Pi}_{\mathsf{Quad}}$ to learn the output $\mathbf{K}^1[\mathbf{K}^2[f]]$. They then evaluate GC_1 to obtain $G(x_1, f(x_2))$

If any point in time, if one of the parties abort, the rest of the parties abort as well. This completes the description of the protocol.

We now argue security. However, we refer the reader to the full version of our paper [2] for more details on the security, correctness and efficiency of this construction.

Security. We consider the following cases. Let S be the set of parties corrupted by the adversary.

$P_1 \in S$ and $P_2 \notin S$. The simulator is defined as follows:

- **Round 1**.
  - Simulating on behalf of P_2: Execute the simulator of $\mathit{\Pi}_{\mathsf{Quad}}$ to obtain the first round messages of $\mathit{\Pi}_{\mathsf{Quad}}$. Generate $R \xleftarrow{\$} \{0,1\}^{|\mathsf{GC}_2|}$. Send the first round messages of $\mathit{\Pi}_{\mathsf{Quad}}$ along with R, intended for the parties in S, to the adversary.
  - Simulating on behalf of parties in $\overline{S} \backslash \{P_2\}$: Execute the simulator of $\mathit{\Pi}_{\mathsf{Quad}}$ to obtain the first round messages of $\mathit{\Pi}_{\mathsf{Quad}}$. Send the first round messages, intended for the parties in S, to the adversary.

  Also, extract the input $\mathbf{K}^1_I$ of P_1 in $\mathit{\Pi}_{\mathsf{Quad}}$ from the first round messages of $\mathit{\Pi}_{\mathsf{Quad}}$.
- **Round 2**. At the end of Round 1, the simulator receives $(f, \widehat{y})$ from the environment.
  - Simulating on behalf of P_2: Execute the simulator Sim_{GC} of the garbling scheme $(\mathsf{Gen}, \mathsf{Garb}, \mathsf{Eval})$; generate $\left(\widehat{\mathsf{GC}_2}, \widehat{\mathbf{K}^2}\right) \leftarrow \mathsf{Sim}_{GC}(1^{\mathbf{k}}, \varphi(U_{x_2}), \widehat{y})$, where $\varphi(U_{x_2})$ is the topology of U_x. Execute the simulator of $\mathit{\Pi}_{\mathsf{Quad}}$[8], with the output of $\mathit{\Pi}_{\mathsf{Quad}}$ set to be $\mathbf{K}^1_I\left[\widehat{\mathbf{K}^2_I} \circ R \oplus \widehat{\mathsf{GC}_2}\right]$, to generate the second round messages of $\mathit{\Pi}_{\mathsf{Quad}}$. Send the second round messages of $\mathit{\Pi}_{\mathsf{Quad}}$, intended for the parties in S, to the adversary.

[8] By the privacy with knowledge of outputs property, the simulator of $\mathit{\Pi}_{\mathsf{Quad}}$ directs the ideal functionality to deliver outputs (of its choice) to honest parties. However, the outer simulator (i.e., the simulator of $\mathit{\Pi}_{\mathsf{DFunc}}$), which is running the simulator of $\mathit{\Pi}_{\mathsf{Quad}}$ as a subroutine, discards these outputs.

  - Simulating on behalf of parties in $\overline{S}\backslash\{P_2\}$: Similar to simulation on behalf of P_2, Execute the simulator of Π_{Quad}, with the output of Π_{Quad} set to be $\mathbf{K}_I^1\left[\widehat{\mathbf{K}_I^2}\circ R\oplus\widehat{\mathsf{GC}_2}\right]$, to generate the second round messages of Π_{Quad}. Send the second round messages of Π_{Quad}, intended for the parties in S, to the adversary.

- **Reconstruction**. Receive the second round messages of Π_{Quad} from the corrupted parties in S. Also receive $\widehat{\mathsf{GC}_1}$ from P_1. Reconstruct the output $\widehat{\mathbf{K}_I^1}$ from the second round messages of Π_{Quad}. Evaluate $\mathsf{Eval}\left(\widehat{\mathsf{GC}_1},\widehat{\mathbf{K}_I^1}\right)$ to obtain $\widehat{b}$. Output $\widehat{b}$.

  If at any point in time, the adversary aborts, the simulator aborts as well.

<u>$P_2\in S$ and $P_1\notin S$.</u> The simulator is defined as follows:

- **Round 1**.
  - Simulating on behalf of P_1: Execute the simulator of Π_{Quad} to obtain the first round messages. Send the messages intended for the parties in S to the adversary.
  - Simulating on behalf of parties in $\overline{S}\backslash P_1$: This is identical to the simulation on behalf of P_1.

  Receive the first round messages of Π_{Quad} from the adversary. Additionally receive $\widehat{R}$ (masked garbled circuit) from P_2. Extract the input $(\widehat{\mathbf{K}_I^2}\circ R)$ of P_2 from the first round messages of Π_{Quad} generated by P_2.
- **Round 2**. At the end of first round, the simulator receives f from the environment. Compute $\mathsf{Eval}\left(\widehat{\mathsf{GC}_2},\widehat{\mathbf{K}_I^2}[f]\right)$ to obtain $\widehat{y}$, where $\widehat{\mathsf{GC}_2}=\widehat{R}\oplus R$. Send $\widehat{y}$ to the ideal functionality to receive $\widehat{b}$.
  - Simulating on behalf of P_1: Execute the simulator Sim_{GC} of the garbling scheme $(\mathsf{Gen},\mathsf{Garb},\mathsf{Eval})$; generate $\left(\widehat{\mathsf{GC}_1},\widehat{\mathbf{K}_I^1}\right)\leftarrow\mathsf{Sim}_{GC}(1^{\mathbf{k}},\varphi(\widehat{G}),\widehat{b})$. Execute the simulator of Π_{Quad}, with the output of Π_{Quad} set to be $\widehat{\mathbf{K}_I^1}$, to generate the second round messages of Π_{Quad}. Send the second round messages of Π_{Quad} along with the simulated garbled circuit $\widehat{\mathsf{GC}_1}$, intended for the parties in S, to the adversary.
  - Simulating on behalf of parties in $\overline{S}\backslash P_1$: Execute the simulator of Π_{Quad}, with the output of Π_{Quad} set to be $\widehat{\mathbf{K}_I^1}$, to generate the second round messages of Π_{Quad}. Send the second round messages of Π_{Quad} intended for the parties in S to the adversary.
- **Reconstruction**. Receive the second round messages of Π_{Quad} from the corrupted parties in S. Reconstruct the output $\widehat{\mathbf{K}_I^1}$ from the second round messages of Π_{Quad}. Evaluate $\mathsf{Eval}\left(\widehat{\mathsf{GC}_1},\widehat{\mathbf{K}_I^1}\right)$ to obtain $\widehat{b'}$. Direct the ideal functionality to deliver the output $\widehat{b'}$ to the honest parties. Output of the simulator is the view of the adversary.

  If any point in time, the adversary aborts, the simulator aborts as well.

$P_1 \notin S$ and $P_2 \notin S$. The simulator is defined as follows:

- **Round 1**.
  - Simulating on behalf of P_1: Execute the simulator of Π_{Quad} to generate the first round messages; send the messages intended for the parties in S to the adversary.
  - Simulating on behalf of P_2: Execute the simulator of Π_{Quad} to generate the first round messages; send the messages intended for the parties in S to the adversary. Also, send a string $R \xleftarrow{\$} \{0,1\}^{|\mathsf{GC}_2|}$.
  - Simulating on behalf of parties in $\overline{S}\backslash\{P_1, P_2\}$: This is identical to the simulation on behalf of P_1.
- **Round 2**. The simulator receives the value $\widehat{b}$ from the ideal functionality.
  - Simulating on behalf of P_1: Execute the simulator Sim_{GC} of $(\mathsf{Gen}, \mathsf{Garb}, \mathsf{Eval})$; compute $\left(\widehat{\mathsf{GC}_1}, \widehat{\mathbf{K}^1}\right) \leftarrow \mathsf{Sim}_{GC}\left(1^{\mathbf{k}}, \varphi(\widehat{G}), \widehat{b}\right)$. Execute the simulator of Π_{Quad}, with the output of Π_{Quad} set to be $\widehat{\mathbf{K}^1}$, to generate the second round messages; send the messages intended for the parties in S to the adversary.
  - Simulating on behalf of P_2: Execute the simulator of Π_{Quad}, with the output of Π_{Quad} set to be $\widehat{\mathbf{K}^1}$, to generate the second round messages; send the messages intended for the parties in S to the adversary.
  - Simulating on behalf of parties in $\overline{S}\backslash\{P_1, P_2\}$: This is identical to the simulation on behalf of P_2.
- **Reconstruction**. Receive the second round messages of Π_{Quad} from the corrupted parties in S. Reconstruct the output $\widehat{\mathbf{K}^1_I}$ from the second round messages of Π_{Quad}. Evaluate $\mathsf{Eval}\left(\widehat{\mathsf{GC}_1}, \widehat{\mathbf{K}^1_I}\right)$ to obtain $\widehat{b'}$. Direct the ideal functionality to deliver the output $\widehat{b'}$ to the honest parties. Output of the simulator is the view of the adversary.

  If any point in time, the adversary aborts, the simulator aborts as well.

□

4 Generalized Conforming Protocols

The notion of conforming protocols was first defined in [22] as an intermediate tool to construct two-round secure MPC from two-round oblivious transfer. Their notion as-is is insufficient to achieve our goal of constructing an information-theoretic multiparty computation protocol secure against malicious adversaries. To get around this, we define the notion of *generalized conforming protocols.*

Syntax. An n-party generalized conforming protocol Φ for an n-party functionality F is specified by the parameters $\left(n, N, \{\Phi_{i,j}\}_{i\in[n],j\in[t+1]}, \mathcal{P}\right)$, where n is the number of parties in the system, N denotes the size of the global state $\mathbf{Z}$, $\Phi_{i,j}$ is a set of actions and $\mathcal{P}$ is a set of $(2 \cdot \binom{n}{2} + n)$ partitions of $[N]$. We denote $\mathcal{P} = (S_1, \ldots, S_n, \{T_{i_1,i_2}\}_{i_1,i_2\in[n],i_1\neq i_2}, U)$. One can think of S_i as the set of locations reserved for private computation for party P_i, T_{i_1,i_2} as the space allocated to party P_{i_1} for communicating private messages to party P_{i_2} and U as the space allocated for storing broadcast messages of each party. A generalized conforming protocol proceeds as follows. Let $x_1, \ldots, x_n$ be the respective inputs of all parties.

- **Pre-processing Phase.** For each $i \in [n]$, party P_i defines $\mathbf{st}_i$ to be the list: $\mathbf{st}_i := \Big(R_k : \forall k \in S_i \bigcup_{i \neq i'} T_{i,i'} \bigcup_{i \neq i'} T_{i',i}\Big)$ where $\forall k \in S_i \bigcup_{i \neq i'} T_{i,i'}$ it samples each bit R_k uniformly at random. Compute an N-sized list $\mathbf{Z}^{i,1}$ as follows:
  - For each $k \in [N]$, initialize $\mathbf{Z}_k^{i,1} = 0$. Here $\mathbf{Z}_k^{i,1}$ denotes the k^{th} bit of $\mathbf{Z}^{i,1}$
  - Compute $\{(z_k, k) : k \in L_i\} \leftarrow \mathsf{Pre}(1^{\mathbf{k}}, i, x_i, \mathbf{st}_i)$ where L_i is a subset of $S_i \bigcup_{i \neq i'} T_{i,i'}$.
  - For every $k \in L_i$, set the k^{th} location $\mathbf{Z}_k^{i,1}$ in $\mathbf{Z}^{i,1}$ to have the value z_k.
  - For each $i' \in [n] \setminus \{i\}$, it sends $(R_k : \forall k \in T_{i,i'})$ to party $P_{i'}$ over private channels.
  - It broadcasts $\mathbf{Z}^{i,1}$ to all other parties.

  We require that there does not exist $k \in [N]$ such that for any $i_1 \neq i_2$, the set output by $\mathsf{Pre}(1^{\mathbf{k}}, i_1, x_{i_1}, \mathbf{st}_{i_1})$ contains $(\cdot, k)$ and the set output by $\mathsf{Pre}(1^{\mathbf{k}}, i_2, x_{i_2}, \mathbf{st}_{i_2})$ also contains $(\cdot, k)$. This means that there is no location in the global state $\mathbf{Z}$ that gets overwritten twice.

  At the end of the pre-processing phase, P_i receives $(R_k : \forall k \in T_{i',i})$ from all other parties $P_{i'}$ $(i' \in [n] \setminus i)$. It includes this as a part of $\mathbf{st}_i$. It retains $\mathbf{st}_i$ as private information.
- **Computation Phase** For each $i \in [n]$, party P_i sets $\mathbf{Z}^1 = \bigoplus_{i-1}^{n} \mathbf{Z}^{i,1}$ For each $j \in [t+1]$, it proceeds as follows:
  - Parse the action $\Phi_{i,j}$ as $(\mathbf{L}_{i,j}^I, \mathbf{C}_{i,j}, \mathbf{L}_{i,j}^O)$.
  - If $j \neq 1$, for $\{(k, z_k)\}_{\forall i' \neq i,\ k \in \mathbf{L}_{i',j-1}^O}$, update k^{th} location in $\mathbf{Z}^{i,j}$ with value z_k. Call the resulting state $\mathbf{Z}^j$.
  - Take as input values in the locations of $\mathbf{Z}^j$ specified by the set $\mathbf{L}_{i,j}^I$ along with $\mathbf{st}_i$, compute $\mathbf{C}_{i,j}$ and update the locations in $\mathbf{Z}^j$ specified by the set $\mathbf{L}_{i,j}^O$. Call the resulting state $\mathbf{Z}^{i,j+1}$.
  - Send all the updated values and locations $\{(k, z_k)\}_{k \in \mathbf{L}_{i,j}^O}$ to all other parties.

  As before, we require that there is no location in $\mathbf{Z}$, where two parties simultaneously write to this location in any given round. At the end of all the rounds, the output of the computation for party P_i is in the last ℓ_i' locations of S_i.

- **Reconstruction.** For every $i \in [n]$, party P_i unmasks the last ℓ_i' locations of S_i to learn the output.

In terms of correctness, we require that at the end of the above protocol, the last ℓ_i' locations of S_i contains masked (y_i), where $F(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$. Since a generalized conforming protocol is a special instance of a secure multiparty computation protocol, the security notions for generalized conforming protocols can be defined analogously.

Definition 7 (CLC Property). *An n-party generalized conforming protocol, specified by the parameters $\left(n, N, \{\Phi_{i,j}\}_{i\in[n],j\in[t+1]}, \mathcal{P}\right)$, for an n-party functionality F satisfies CLC property if the following holds: every $\Phi_{i,j}$ can be parsed as $(\mathbf{L}^{I}_{i,j} = \mathbf{L}^{I\rightarrow}_{i,j} \cup \mathbf{L}^{I\leftarrow}_{i,j},\ \mathbf{C}_{i,j},\ \mathbf{L}^{O}_{i,j} = \mathbf{L}^{O\rightarrow}_{i,j} \cup \mathbf{L}^{O\leftarrow}_{i,j})$. We require that $\mathbf{C}_{i,j}$, for every $i \in [n], j \in [t+1]\backslash\{1\}$ (that is, all rounds except the first), is defined as follows: it takes as input values in the locations of $\mathbf{Z}$ specified by the locations $\mathbf{L}^{I}_{i,j} = \mathbf{L}^{I\rightarrow}_{i,j} \cup \mathbf{L}^{I\leftarrow}_{i,j}$ and state $\mathbf{st}_i$,*

- **Copy Operation**: *For every $k \in \mathbf{L}^{I\rightarrow}_{i,j}$, there exists a unique $k' \in \mathbf{L}^{I\leftarrow}_{i,j} \subset S_i$, copy $z_k \oplus R_k \oplus R_{k'}$ to $(k')^{th}$ location in $\mathbf{Z}$, where z_k is the value in the k^{th} location of $\mathbf{Z}$. Note that $R_k, R_{k'}$ are values in the list $\mathbf{st}_i$ and hence, $\mathbf{L}^{I\rightarrow}_{i,j} \subset U \bigcup_{i'\in[n]\backslash\{i\}} T_{i,i'} \bigcup_{i'\in[n]\backslash\{i\}} T_{i',i}$.*
- **Local Computation**: *Take as input a set of values in $\mathbf{Z}$, indexed by a subset of S_i, $\mathbf{st}_i$, and compute a polynomial-sized circuit on these values. The output of this computation is written to a subset of locations, indexed by S_i, in $\mathbf{Z}$.*
- **Copy Operation**: *For every $k' \in \mathbf{L}^{O\rightarrow}_{i,j} \subset S_i$, there exists a unique $k \in \mathbf{L}^{O\leftarrow}_{i,j}$, copy $z_{k'} \oplus R_k \oplus R_{k'}$ to k^{th} location in $\mathbf{Z}$, where z_k is the value in the k^{th} location of $\mathbf{Z}$. As before, $R_k, R_{k'}$ are values in the list $\mathbf{st}_i$ and hence, $\mathbf{L}^{O\leftarrow}_{i,j} \subset U \bigcup_{i'\in[n]\backslash\{i\}} T_{i,i'}$.*

For the first round, we require $\mathbf{C}_{i,1}$ to be defined as follows: it takes as input $\mathbf{Z}^1$, computes a circuit $\widehat{\mathbf{C}}_{i,1}$ on $\mathbf{Z}^1$ to obtain $\{v_k\}_{k\in\mathbf{L}^{O}_{i,1}}$ and finally, it updates the k^{th} location in $\mathbf{Z}^{i,2}$ with the value $z_k = v_k \oplus R_k$ for every $k \in \mathbf{L}^{O}_{i,1}$.

Lemma 3. *Let $n, \ell_1, \ell'_1, \ldots, \ell_n, \ell'_n > 0$. Consider an n-party functionality $F : \{0,1\}^{\ell_1} \times \cdots \times \{0,1\}^{\ell_n} \rightarrow \{0,1\}^{\ell'_1} \times \cdots \times \{0,1\}^{\ell'_n}$ computable by a depth-d circuit of size s. There is a maliciously secure t-round generalized conforming protcol for F, for some constant t, satisfying CLC property with perfect security in the honest majority setting. Moreover, the next message function of every party can be implemented by a circuit of depth $O(d + \log(s))$ and size $s^c 2^{c\cdot(d+\log(s))}$, for some constant c.*

We defer the proof of this lemma to the full-version of our paper [2].

5 Two-round MPC over Broadcast and P2P: Security with Abort

In this section, we show how to construct a two-round MPC in the honest majority setting and satisfying statistical malicious security.

Lemma 4. *Let $n, \ell_1, \ell_1', \ldots, \ell_n, \ell_n' > 0$. Consider an n-party functionality $F : \{0,1\}^{\ell_1} \times \cdots \times \{0,1\}^{\ell_n} \rightarrow \{0,1\}^{\ell_1'} \times \cdots \times \{0,1\}^{\ell_n'}$ computable by a depth-d circuit of size s.*

Fix a statistical security parameter $\boldsymbol{k} > 0$. There is a malicious two-round MPC protocol for F with $\mathsf{negl}(\boldsymbol{k})$-statistical security in the honest majority setting, for some negligible function negl. Moreover, the computational complexity of this protocol is polynomial in s and exponential in d.

Tools. We use the following ingredients in our construction.

- t-round Generalized Conforming protocol for F, guaranteed by Lemma 3. Denote this by Π_{GConf}. Let Π_{GConf} be parameterized by $\left(n, N, \{\Phi_{i,j}\}_{i\in[n], j \in [t+1]}, \mathcal{P}\right)$.
- Delayed-function two-round secure n-party MPC for quadratic polynomials, as guaranteed by Lemma 1.
- Delayed-function two-round secure n-party MPC for specialized two-input functionalities, as guaranteed by Lemma 2.
- Information-theoretic garbling scheme $(\mathsf{Gen}, \mathsf{Garb}, \mathsf{Eval})$.

Construction. We describe a two-round secure MPC protocol for F below.

Round 1.

- **Generation of Initial Global State**: For every $i \in [n]$, the i^{th} party computes the pre-processing phase of Π_{GConf}. In particular it does the following: it defines $\mathbf{st}_i := (R_k : \forall k \in S_i \bigcup_{i \neq i'} T_{i,i'} \bigcup_{i \neq i'} T_{i',i})$, where $\forall k \in S_i \bigcup_{i \neq i'}$, it samples the bit R_k uniformly at random. It computes $\mathsf{Pre}(1^{\mathbf{k}}, i, x_i, \mathbf{st}_i)$ to obtain the set $\{(z_k, k) : k \in L_i\}$. It computes a N-sized list $\mathbf{Z}^{i,1}$ as follows: initialize $\mathbf{Z}^{i,1}$ to consist of only zeroes. It sets the k^{th} location in $\mathbf{Z}^{i,1}$ to have the value z_k. Broadcast $\mathbf{Z}^{i,1}$ and sends $(R_k : \forall k \in T_{i,i'})$ to party $P_{i'}$ for each $i' \in [n] \setminus i$ over a private channel.

- **Generation of Garbling Wire Labels**: $(\mathsf{gk}_{i,1}, \mathbf{K}_{i,1}) \leftarrow \mathsf{Gen}(1^{\mathbf{k}}, 1^L, 1^d)$, where L is the number of leaves and d is the depth of the formula in Figure 1a.

- For every $j \in [t+1] \backslash \{1\}$, the i^{th} party computes the following:
  - $(\mathsf{gk}_{i,j}[\mathsf{copy1}], \mathbf{K}_{i,j}[\mathsf{copy1}]) \leftarrow \mathsf{Gen}(1^{\mathbf{k}}, 1^L, 1^d)$, where L is the number of leaves and d is the depth of the formula in Figure 1b.
  - $(\mathsf{gk}_{i,j}[\mathsf{copy2}], \mathbf{K}_{i,j}[\mathsf{copy2}]) \leftarrow \mathsf{Gen}(1^{\mathbf{k}}, 1^L, 1^d)$, where L is the number of leaves and d is the depth of the formula in Figure 2b.

- **First Round Messages of Delayed-Function MPC for Quadratic Polynomials**: All the parties participate in $O(n^3 t)$ executions of delayed-function two-round secure n-party MPC for quadratic polynomials, as guaranteed by Lemma 1. Each of these instantiations are denoted as follows:

- For every $i_1, i_2 \in [n]$ and $i_1 \neq i_2$, the input of the i^{th} party in $\mathit{\Pi}_{\mathsf{Quad}}[i_1, i_2, 1, 1]$ is the following:
  - $*$ If $i = i_1$ then the i^{th} party inputs $\{v_k\}_{k \in \mathbf{L}^O_{i_1,1}}$, $\{R_k\}_{k\mathbf{L}^O_{i_1,1}}$ [9], where $\{v_k\}_{k \in \mathbf{L}^O_{i_1,1}}$ is the output of circuit $\widehat{C}_{i_1,1}$[10] on $\mathbf{Z}^1$, as defined in the definition 7.
  - $*$ If $i = i_2$ then the i^{th} party inputs $\mathbf{K}_{i_2,2}[\mathsf{copy1}]$.
  - $*$ If $i \neq i_1, i \neq i_2$ then the i^{th} party doesn't have any input.

  Denote $\mathit{\Pi}_{\mathsf{Quad}}[i_1, i_2, 1, 1].\mathsf{msg}_{1,i_4 \rightarrow i_5}$ to be the first round message of $\mathit{\Pi}_{\mathsf{Quad}}[i_1, i_2, 1j, 1]$ sent by the $(i_4)^{th}$ party to the $(i_5)^{th}$ party. We don't require that i_4 or i_5 be distinct from i_1, i_2. We define similar notation for the other $\mathit{\Pi}_{\mathsf{Quad}}$ instantiations. Let the total randomness used by the i^{th} party in all these executions be R^i_Q.

- $\{\mathit{\Pi}_{\mathsf{Quad}}[i_1, i_2, i_3, j, 1]\}_{i_1,i_3 \in [n], i_1 \neq i_3, i_2 \in [n+1] \;\; j \in [t+1] \setminus \{1\}}$
  For $i_1, i_3 \in [n], i_1 \neq i_3, i_2 \in [n+1], j \in [t+1] \setminus \{1\}$, the input of the i^{th} party in $\mathit{\Pi}_{\mathsf{Quad}}[i_1, i_2, i_3, j, 1]$ is the following:
  - $*$ If $i = i_1$, then the i^{th} party inputs $\{R_k\}_{k \in S_{i_1}}$
  - $*$ If $i = i_1 = i_2$, then the i^{th} party additionally inputs $\{\{R_{k'}\}_{k \in T_{i_1,i'}}\}_{i' \in [n] \setminus \{1\}}$.
  - $*$ If $i = i_2 \neq i_1$, then the i^{th} party inputs $\{R_{k'}\}_{k' \in T_{i_2,i_1}}$.
    If $i_2 = n+1$, then party P_{i_2} has no input. This corresponds to the copy operations from locations in U to locations in S_{i_1}.
  - $*$ If $i = i_3$ then the i^{th} party inputs $\mathbf{K}_{i_3,j}[\mathsf{local}]$
  - $*$ If $i \neq i_1, i \neq i_2, i \neq i_3$ then the i^{th} party doesn't have any input.

- $\{\mathit{\Pi}_{\mathsf{Quad}}[i_1, i_2, j, 2]\}_{i_1,i_2 \in [n], i_1 \neq i_2, \;\; j \in [t] \setminus \{1\}}$
  For $i_1, i_2 \in [n], i_1 \neq i_2, j \in [t] \setminus \{1\}$, the input of the i^{th} party in $\mathit{\Pi}_{\mathsf{Quad}}[i_1, i_2, j, 1]$ is the following:
  - $*$ If $i = i_1$, then the i^{th} party inputs $\{R_{k'}\}_{k' \in S_{i_1}}$, $\{R_k\}_{k \in U \bigcup_{i' \in [n] \setminus \{i_1\}} T_{i_1,i'}}$
  - $*$ If $i = i_2$ then the i^{th} party inputs $\mathbf{K}_{i_2,j+1}[\mathsf{copy1}]$
  - $*$ If $i \neq i_1, i \neq i_2$ then the i^{th} party doesn't have any input.

[9] Recall that $\mathbf{L}i_1, 1^O$ consists of a subset of locations in $S_{i_1}, \bigcup_{i' \in [n] \setminus \{i_1\}} T_{i_1,i'}$ and U and the locations in U are not a part of $\mathbf{st}_{i_1}$. But since R_k is not a part of $\mathbf{st}_{i'}$ for any $k \in U$ and $i' \in [n]$. Hence this is equivalent to every party setting $R_k = 0$ for all $k \in U$.

[10] Note that the only values from $\mathbf{Z}^1$ that $\widehat{C}_{i_1,1}$ computes on are known to P_{i_1} in the first round itself. Hence even if it does not know the entire value of $\mathbf{Z}^1$ in the first round, values $\{v_k\}_{k \in \mathbf{L}^O_{i_1,1}}$ can still be computed.

The functionalities associated with each of these protocols are determined in the second round.

– **First Round Messages of Delayed-Function MPC for Two-Input Functionalities**: All the parties participate in $O(n^2t)$ executions of delayed-function two-round secure n-party MPC, as guaranteed by Lemma 2. Denote these instantiations to be $\{\mathit{\Pi}_{\mathsf{DFunc}}[i_1, i_2, j]\}_{i_1,i_2 \in [n], j \in [t+1] \setminus \{1\}}$. For every $i_1, i_2 \in [n]$ and $i_1 \neq i_2, j \in [t+1] \setminus \{1\}$, the input of i^{th} party in $\mathit{\Pi}_{\mathsf{DFunc}}[i_1, i_2, j]$ is the following:
  - If $i = i_1$ then the i^{th} party inputs $\mathbf{K}_{i_1,j}[\mathsf{copy2}]$.
  - If $i = i_2$ then the i^{th} party inputs $\{R_k\}_{k \in S_{i_2}}$.
  - If $i \neq i_1, i \neq i_2$ then the i^{th} party doesn't have any input.

  Denote $\mathit{\Pi}_{\mathsf{DFunc}}[i_1, i_2, j].\mathsf{msg}_{1,i \to i''}$ to be the first message of $\mathit{\Pi}_{\mathsf{DFunc}}[i', j]$ sent by the i^{th} party to $(i'')^{th}$ party. Let the total randomness used by the i^{th} party in all the executions be R^i_{DF}.

Round 2.

– **Compute Joint Global State**: All the parties compute $\mathbf{Z}^1 = \bigoplus_{i=1}^n \mathbf{Z}^{i,1}$.

– **Updates Private State**: It updates $\mathbf{st}_i$ to include $(R_k : \forall k \in T_{i',i})$ received from party $P_{i'} (\forall i' \in [n] \setminus i)$ in the first round.

– **Generate Input Wire Labels for First Garbled Circuit**: The i^{th} party computes $(\mathsf{GC}_{i,1}, \mathbf{K}_{i,1}) \leftarrow \mathsf{Garb}(\mathsf{gk}_{i,1}, C_{i,1})$, where $C_{i,1}$ is defined in Figure 1a. Let $\mathbf{K}_{i,1}\left[\mathbf{Z}^1\right]$ be the set of wire keys corresponding to the input $\mathbf{Z}^1$.

– **Generate Garbled Circuits for every round of Generalized Conforming Protocol**: For every $j \in [t+1] \setminus \{1\}$, the i^{th} party computes:
  - $(\mathsf{GC}_{i,j}[\mathsf{copy1}], \mathbf{K}_{i,j}[\mathsf{copy1}]) \leftarrow \mathsf{Garb}(\mathsf{gk}_{i,j}[\mathsf{copy1}], C_{i,j}[\mathsf{copy1}])$, where $C_{i,j}[\mathsf{copy1}]$ is defined in Figure 1b.
  - $(\mathsf{GC}_{i,j}[\mathsf{local}], \mathbf{K}_{i,j}[\mathsf{local}]) \leftarrow \mathsf{Garb}(\mathsf{gk}_{i,j}[\mathsf{local}], C_{i,j}[\mathsf{local}])$, where $C_{i,j}[\mathsf{local}]$ is defined in Figure 2a.
  - $(\mathsf{GC}_{i,j}[\mathsf{copy2}], \mathbf{K}_{i,j}[\mathsf{copy2}]) \leftarrow \mathsf{Garb}(\mathsf{gk}_{i,j}[\mathsf{copy2}], C_{i,j}[\mathsf{copy2}])$, where $C_{i,j}[\mathsf{copy2}]$ is defined in Figure 2b.

– The i^{th} party broadcasts the following message:
$\Big(\mathsf{GC}_{i,1},\ \mathbf{K}_{i,1}\left[\mathbf{Z}^1\right], \{\mathsf{GC}_{i,j}[\mathsf{copy1}], \mathsf{GC}_{i,j}[\mathsf{local}], \mathsf{GC}_{i,j}[\mathsf{copy2}]\}_{j \in [t+1]}\Big)$

  Evaluation. To compute the output of the protocol, each party P_i does the following:
  - For each $i' \in [n]$, let $\mathbf{K}_{i',1}[\mathbf{Z}^1]$ be the labels received from party $P_{i'}$ at the end of round 2.
  - Obtain For each $i' \in [n]$, compute $\mathsf{Eval}(\mathsf{GC}_{i',1}, \mathbf{K}_{i',1}[\mathbf{Z}^1])$ to obtain labels in $\mathbf{K}_{i',2}[\mathsf{copy1}]$ corresponding to $\mathbf{Z}^{i',2}[\mathsf{copy1}]$ and second round messages

$\{\mathit{\Pi}_{\mathsf{Quad}}[i_1, i_2, 1, 1].\mathsf{msg}_{2,i' \to i''}\}_{i_1,i_2,i',i'' \in [n], i_1 \neq i_2}$. Use these second round messages to reconstruct the remaining labels in $\mathbf{K}_{i',2}[\mathsf{copy1}]$ corresponding to $\{(k, z_k)\}_{\forall i'' \neq i',\ k \in \mathbf{L}^{O}_{i,1}}$.

- For each j from 2 to $(t+1)$ do the following:
  - For each $i' \in [n]$, compute $\mathsf{Eval}(\mathsf{GC}_{i',j}[\mathsf{copy1}], \mathbf{K}_{i',j}[\mathsf{copy1}][\mathbf{Z}^{i',j}[\mathsf{copy1}] || \{(k, z_k)\}_{\forall i'' \neq i',\ k \in \mathbf{L}^{O\leftarrow}_{i'',j-1}}])$ (if $j = 2, \mathbf{L}^{O\leftarrow}_{i'',j-1} = \mathbf{L}^{O}_{i'',j-1}$) to obtain labels in $\mathbf{K}_{i',j}[\mathsf{local}]$ corresponding to $\mathbf{Z}^{i',j}[\mathsf{local}]$ and second round messages $\{\mathit{\Pi}_{\mathsf{Quad}}[i_1, i_2, i_3, j, 1].\mathsf{msg}_{2,i' \to i''}\}_{i_1,i_3,i',i'' \in [n], i_1 \neq i_3, i_2 \in [n+1]}$.
  - Use these second round messages to reconstruct the remaining labels in $\mathbf{K}_{i',j}[\mathsf{local}]$ corresponding to $\{(k, z_k)\}_{\forall i'' \neq i',\ k \in \mathbf{L}^{I\leftarrow}_{i'',j}}$.
  - For each $i' \in [n]$, compute $\mathsf{Eval}(\mathsf{GC}_{i',j}[\mathsf{local}], \mathbf{K}_{i',j}[\mathsf{local}][\mathbf{Z}^{i',j}[\mathsf{local}] || \{(k, z_k)\}_{\forall i'' \neq i',\ k \in \mathbf{L}^{I\leftarrow}_{i'',j}}])$ to obtain labels in $\mathbf{K}_{i',j}[\mathsf{copy2}]$ corresponding to $\mathbf{Z}^{i',j}[\mathsf{copy2}]$ and second round messages $\{\mathit{\Pi}_{\mathsf{DFunc}}[i_1, i_2, j, 1].\mathsf{msg}_{2,i' \to i''}\}_{i_1,i_2,i',i'' \in [n], i_1 \neq i_2}$.
  - Use these second round messages to reconstruct the remaining labels in $\mathbf{K}_{i,j}[\mathsf{copy2}]$ corresponding to $\{(k, z_k)\}_{\forall i'' \neq i',\ k \in S_{i''}}$.
  - For each $i' \in [n]$, if $(j \neq t{+}1)$, compute $\mathsf{Eval}(\mathsf{GC}_{i',j}[\mathsf{copy2}], \mathbf{K}_{i',j}[\mathsf{copy2}]$ $[\mathbf{Z}^{i',j}[\mathsf{copy2}] || \{(k, z_k)\}_{\forall i'' \neq i',\ k \in S_{i''}}])$ to obtain labels in $\mathbf{K}_{i',j+1}[\mathsf{copy1}]$ corresponding to $\mathbf{Z}^{i',j+1}[\mathsf{copy1}]$ and second round messages $\{\mathit{\Pi}_{\mathsf{Quad}}[i_1, i_2 j, 1].\mathsf{msg}_{2,i' \to i''}\}_{i_1,i_2,i',i'' \in [n], i_1 \neq i_2}$.
  - Use these second round messages to reconstruct the remaining labels in $\mathbf{K}_{i',j}[\mathsf{local}]$ corresponding to $\{(k, z_k)\}_{\forall i'' \neq i',\ k \in \mathbf{L}^{O\leftarrow}_{i'',j-1}}$.
  - If $j = t + 1$ compute $\mathsf{Eval}(\mathsf{GC}_{i',j}[\mathsf{copy2}], \mathbf{K}_{i',j}[\mathsf{copy2}][\mathbf{Z}^{i',j}[\mathsf{copy2}] || \{(k, z_k)\}_{\forall i'' \neq i',\ k \in S_{i''}}])$ to obtain $\mathbf{Z}_{fin}$.
- Use $\mathbf{st}_i$ to unmask the last ℓ_i locations of S_i in $\mathbf{Z}_{fin}$ to compute the output.

We defer the security of this protocol to the full-version of our paper [2].

6 Two-Round MPC over P2P: Security with Selective Abort

In this section we describe a general compiler to obtain a two-round information-theoretic MPC protocol satisfying malicious security with selective abort over point-point channels from any two-round information-theoretic MPC satisfying security with abort against malicious adversaries.

Input: $\mathbf{Z}^1$
Hardwired Values: action $\Phi_{i,1} = (\mathbf{L}^I_{i,1}, \mathbf{C}_{i,1}, \mathbf{L}^O_{i,1})$, state $\mathbf{st}_i$, wire labels $\mathbf{K}_{i,2}[\mathsf{copy1}]$

- Parse $\Phi_{i,1} = (\mathbf{L}^I_{i,1}, \mathbf{C}_{i,1}, \mathbf{L}^O_{i,1})$
- Compute $\widehat{\mathbf{C}}_{i,1}$, where $\widehat{\mathbf{C}}_{i,1}$ is the circuit associated with $\mathbf{C}_{i,1}$ (see Definition 7), on the values in $\mathbf{Z}^1$ indexed by $\mathbf{L}^I_{i,1}$ along with $\mathbf{st}_i$. The output of the computation is written to locations in $\mathbf{Z}^1$ indexed by $\mathbf{L}^O_{i,1}$. Call the resulting state $\mathbf{Z}^{i,2}[\mathsf{copy1}]$.
- The input to $C_{i,2}[\mathsf{copy1}]$ is of the form $\left(\mathbf{Z}^{i,2}[\mathsf{copy1}], \{(k, z_k)\}_{\forall i' \neq i,\ k \in \mathbf{L}^O_{i',1}}\right)$. Thus, the wire labels $\mathbf{K}_{i,2}[\mathsf{copy1}]$ can be divided into two parts: the first part corresponds to $\mathbf{Z}^{i,2}[\mathsf{copy1}]$ and the second part corresponds to $\{(k, z_k)\}_{\forall i' \neq i,\ k \in \mathbf{L}^O_{i',1}}$.
- For every i_1, i_2 with $i_1 \neq i_2$, compute the second round messages of $\Pi_{\mathsf{Quad}}[i_1, i_2, 1, 1]$. The n-party functionality associated with $\Pi_{\mathsf{Quad}}[i_1, i_2, 1, 1]$ is $Q_{i_1,i_2,1,1}$, defined below.

  The $(i_1)^{th}$ party has input $\{v_k\}_{k \in \mathbf{L}^O_{i,1}}$, $\{R_k\}_{k \in S_i \bigcup_{i' \in [n] \setminus \{i_1\}} T_{i_1,i'}}$, the $(i_2)^{th}$ party has input $\mathbf{K}_{i_2,2}[\mathsf{copy1}]$, the rest of the parties don't have any input and the output of this function are the labels in $\mathbf{K}_{i_2,2}[\mathsf{copy1}]$ corresponding to $\{(k, R_k \oplus v_k)\}$, for every location $k \in \mathbf{L}^O_{i_1,1}$; recall that $\mathbf{L}^O_{i_1,1}$ is the set of the locations written to at the end of first round in the conforming protocol.

  Output the second round messages of all these protocols. Also, output the labels in $\mathbf{K}_{i,2}[\mathsf{copy1}]$ with respect to updated state $\mathbf{Z}^{i,2}[\mathsf{copy1}]$.

(a) Description of $C_{i,1}$

Input: $\left(\mathbf{Z}^{i,j}[\mathsf{copy1}],\ \{(k, z_k)\}_{\forall i' \neq i,\ k \in \mathbf{L}^{O\leftarrow}_{i',j-1}}\right)$ (If $j = 2$, then $\mathbf{L}^{O\leftarrow}_{i',1} = \mathbf{L}^O_{i',1}$)
Hardwired Values: action $\Phi_{i,j} = (\mathbf{L}^I_{i,j}, \mathbf{C}_{i,j}, \mathbf{L}^O_{i,j})$, state $\mathbf{st}_i$, wire labels $\mathbf{K}_{i,j}[\mathsf{local}]$.

- For every $i' \neq i$, every $k \in \mathbf{L}^O_{i',j-1}$, update the k^{th} location in $\mathbf{Z}^{i,j}[\mathsf{copy1}]$ with the value z_k. Call the resulting state $\mathbf{Z}^j[\mathsf{copy1}]$.
- Compute the first copy operation of $\Phi_{i,j}$ on the global state $\mathbf{Z}^j[\mathsf{copy1}]$. Call the resulting state $\mathbf{Z}^{i,j}[\mathsf{local}]$.
- The input to $C_{i,j}[\mathsf{local}]$ is of the form $\left(\mathbf{Z}^{i,j}[\mathsf{local}], \{(k, z_k)\}_{\forall i' \neq i,\ k \in \mathbf{L}^{I\leftarrow}_{i',j}}\right)$ Thus, the wire labels $\mathbf{K}_{i,j}[\mathsf{local}]$ can be divided into two parts: the first part corresponds to $\mathbf{Z}^{i,j}[\mathsf{local}]$ and the second part corresponds to $\{(k, z_k)\}_{\forall i' \neq i,\ k \in \mathbf{L}^{I\leftarrow}_{i',j}}$.
- For every $i_1, i_3 \in [n]$ with $i_1 \neq i_3$ and $i_2 \in [n+1]$, compute the second round messages of $\Pi_{\mathsf{Quad}}[i_1, i_2, i_3, j, 1]$. The n-party functionality associated with $\Pi_{\mathsf{Quad}}[i_1, i_2, i_3, j, 1]$ is $Q_{i_1,i_2,i_3,j,1}$, defined below.

  The $(i_1)^{th}$ party has input $\{R_{k'}\}_{k' \in S_{i_1}}$, the $(i_2)^{th}$ party has input $\{R_k\}_{k \in T_{i_2,i_1}}$ (if $i_1 = i_2$,, the i_1^{th} party additionally has input $\{\{R_k\}_{k \in T_{i_1,i'}}\}_{i' \in [n] \setminus \{i_1\}}$ and if $i_2 = n+1$, the $(i_2)^{th}$ party has no input), the $(i_3)^{th}$ party has input $\mathbf{K}_{i_3,j}[\mathsf{local}]$, the rest of the parties don't have any input and the output of this function are the labels in $\mathbf{K}_{i_3,j}[\mathsf{local}]$ corresponding to $\{(k',\ R_{k'} \oplus R_k \oplus z_k)\}$, for every location $k \in \mathbf{L}^{I\rightarrow}_{i_1,j}$ and $k' \in \mathbf{L}^{I\leftarrow}_{i_1,j}$ such that k' is the unique location associated with k as guaranteed by Definition 7.

  Output the second round messages of all these protocols. Also, output the labels in $\mathbf{K}_{i,j}[\mathsf{local}]$ with respect to updated state $\mathbf{Z}^{i,j}[\mathsf{local}]$.

(b) Description of $C_{i,j}[\mathsf{copy1}]$, for $j > 1$.

Fig. 1

Input: $\left(\mathbf{Z}^{i,j}[\mathsf{local}],\ \{(k, z_k)\}_{\forall i' \neq i,\ k \in \mathbf{L}^{I\leftarrow}_{i',j}}\right)$

Hardwired Values: action $\Phi_{i,j} = (\mathbf{L}^{I}_{i,j}, \mathbf{C}_{i,j}, \mathbf{L}^{O}_{i,j})$, state $\mathbf{st}_i$, wire labels $\mathbf{K}_{i,j}[\mathsf{copy2}]$.

- For every $i' \neq i$, for every $k \in \mathbf{L}^{I\leftarrow}_{i',j}$, update the k^{th} location in $\mathbf{Z}^{i',j}[\mathsf{local}]$ with the value z_k. Call the resulting state $\mathbf{Z}^{j}[\mathsf{local}]$.
- Compute the local operation of $\Phi_{i,j}$ on the global state $\mathbf{Z}^{j}[\mathsf{local}]$. Call the resulting state $\mathbf{Z}^{i,j}[\mathsf{copy2}]$.
- The input to $C_{i,j}[\mathsf{copy2}]$ is of the form $\left(\mathbf{Z}^{i,j}[\mathsf{copy2}], \{(k, z_k)\}_{\forall i' \neq i,\ k \in S_{i'}}\right)$. Thus, the wire labels $\mathbf{K}_{i,j}[\mathsf{copy2}]$ can be divided into two parts: the first part corresponding to $\mathbf{Z}^{i,j}[\mathsf{copy2}]$ and the second part corresponding to $\{(k, z_k)\}_{\forall i' \neq i,\ k \in S_{i'}}$.
- For every i_1, i_2 with $i_1 \neq i_2$, compute the second round messages of $\Pi_{\mathsf{DFunc}}[i_1, i_2, j]$. The n-party functionality associated with $\Pi_{\mathsf{DFunc}}[i_1, i_2, j]$ is $DF_{i_1,i_2,j}$, defined below.

  > The i_1^{th} party has the input $\mathbf{K}_{i_1,j}[\mathsf{copy2}]$, the i_2^{th} party has the input $\{R_k\}_{k \in S_{i_2}}$, the rest of the parties don't have any input and the output of the function is computed as follows: compute the local operation of $\Phi_{i_2,j}$ on the global state $\mathbf{Z}^{j}[\mathsf{local}]$ and the output of the function are the labels in $\mathbf{K}_{i_1,j}[\mathsf{copy2}]$ corresponding to $\{(k, z_k)\}$, for every location $k \in S_{i_2}$.

  Output the second round messages of all these protocols. Also, output the labels in $\mathbf{K}_{i,j}[\mathsf{copy2}]$ with respect to updated state $\mathbf{Z}^{i,j}[\mathsf{copy2}]$.

(a) Description of $C_{i,1}[\mathsf{local}]$

Input: $\left(\mathbf{Z}^{i,j}[\mathsf{copy2}], \{(k, z_k)\}_{\forall i' \neq i,\ k \in S_{i'}}\right)$

Hardwired Values: action $\Phi_{i,j} = (\mathbf{L}^{I}_{i,j}, \mathbf{C}_{i,j}, \mathbf{L}^{O}_{i,j})$, state $\mathbf{st}_i$, wire labels $\mathbf{K}_{i+1,j}[\mathsf{copy1}]$.

- For every $i' \neq i$, for every $k \in S_{i'}$, update the k^{th} location in $\mathbf{Z}^{i',j}[\mathsf{copy2}]$ with the value v_k. If $j \neq t+1$, call the resulting state $\mathbf{Z}^{j}[\mathsf{copy2}]$, otherwise call the resulting state $\mathbf{Z}_{fin}$.
- If $j = t+1$, output $\mathbf{Z}_{fin}$, else continue to next step.
- Compute the second copy operation of $\Phi_{i,j}$ on the global state $\mathbf{Z}^{j}[\mathsf{copy2}]$. Call the resulting state $\mathbf{Z}^{i,j+1}[\mathsf{copy1}]$.
- The input to $C_{i,j+1}[\mathsf{copy1}]$ is of the form $\left(\mathbf{Z}^{i,j+1}[\mathsf{copy1}], \{(k, z_k)\}_{\forall i' \neq i,\ k \in \mathbf{L}^{O}_{i',j}\leftarrow}\right)$. Thus, the wire labels $\mathbf{K}_{i,j+1}[\mathsf{copy1}]$ can be divided into two parts: the first part corresponding to $\mathbf{Z}^{i,j+1}[\mathsf{copy1}]$ and the second part corresponding to $\{(k, z_k)\}_{\forall i' \neq i, \mathbf{L}^{O}_{i',j}\leftarrow}$.
- For every i_1, i_2 with $i_1 \neq i_2$, compute the second round messages of $\Pi_{\mathsf{Quad}}[i_1, i_2, j, 2]$. The n-party functionality associated with $\Pi_{\mathsf{Quad}}[i_1, i_2, j, 2]$ is $Q_{i_1,i_2,j,2}$, defined below.

  > The $(i_1)^{th}$ party has input $\mathbf{st}_i$, the $(i_2)^{th}$ party has input $\mathbf{K}_{i_2,j+1}[\mathsf{copy1}]$, the rest of the parties don't have any input and the output of this function are the labels in $\mathbf{K}_{i_2,j+1}[\mathsf{copy1}]$ corresponding to $\{(k,\ R_k \oplus R_{k'} \oplus z_k)\}$, for every location $k \in \mathbf{L}^{O\leftarrow}_{i_1,j}$ and $k' \in \mathbf{L}^{O\rightarrow}_{i_1,j}$ such that k' is the unique location associated with k as guaranteed by Definition 7.

  Output the second round messages of all these protocols. Also, output the labels in $\mathbf{K}_{i,j+1}[\mathsf{copy1}]$ with respect to updated state $\mathbf{Z}^{i,j+1}[\mathsf{copy1}]$.

(b) Description of $C_{i,j}[\mathsf{copy2}]$.

Fig. 2

Theorem 3. *There exists a general information theoretic compiler that transforms any two round maliciously secure MPC protocol (whose second round messages are computable in NC*^{1}*) over broadcast and private channels that achieves security with abort into a two-round protocol over private channels that achieves selective security with abort against malicious adversaries.*

Building Blocks. We use the following ingredients in our construction. A two-round MPC $\mathit{\Pi}$, in the honest majority setting satisfying perfect malicious security. We additionally want the second round next-message function of each party in this protocol to be implementable using an NC^1 circuit from Section 5. Information-theoretic garbling scheme $(\mathsf{Gen}, \mathsf{Garb}, \mathsf{Eval})$.

Protocol. Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of parties in the protocol. Let $\{x_1, \ldots, x_n\}$ be their respective inputs and $r_1, \ldots, r_n$ be their respective randomness used in the underlying protocol $\mathit{\Pi}$. Let $\mathbf{k}$ be the statistical security parameter.

Round 1. For each $i \in [n]$, party P_i does the following in the first round.

- Compute the first round messages of $\mathit{\Pi}$.
  $(\mathit{\Pi}.\mathsf{msg}_{1,i\to B}, \{\mathit{\Pi}.\mathsf{msg}_{1,i\to j}\}_{j\in[n]}) \coloneqq \mathit{\Pi}_1(1^{\mathbf{k}}, i, x_i; r_i)$ where $\mathit{\Pi}_1$ is the first round next message function of the protocol $\mathit{\Pi}$.
  $\mathit{\Pi}.\mathsf{msg}_{1,i\to B}$ is the message that is broadcast by P_i in the first round of $\mathit{\Pi}$ and $\mathit{\Pi}.\mathsf{msg}_{1,i\to j}$ is the message that is sent to party P_j over a private channel.
- Computes $(\mathsf{gk}_i, \mathbf{K}_i) \leftarrow \mathsf{Gen}(1^{\mathbf{k}}, 1^L, 1^d)$ where L is the number of leaves and d is the depth of the second round next message function of $\mathit{\Pi}$ as defined in the next round. We parse $\mathbf{K}_i$ as $(K^0_{i,1}, K^1_{i,1}, \ldots, K^0_{i,L}, K^1_{i,L})$
- Compute shares $\{\{K^{b,j}_{i,\ell}\}_{j\in[n]}\}_{\ell\in[L],b\in\{0,1\}}$ such that for each $\ell \in [L]$ and $b \in \{0,1\}$, $K^b_{i,\ell} \coloneqq \bigoplus_{j\in[n]} K^{b,j}_{i,\ell}$
- For each $j \in [n] \setminus \{i\}$, it sends $(\mathit{\Pi}.\mathsf{msg}_{1,i\to B}, \mathit{\Pi}.\mathsf{msg}_{1,i\to j}, \{K^{b,j}_{i,\ell}\}_{\ell\in[L],b\in\{0,1\}})$ to party P_j over a private channel.

Round 2. For each $i \in [n]$, party P_i does the following.

- Computes a garbled circuit as follows:
  $\mathsf{GC}_i \leftarrow \mathsf{Garb}(\mathsf{gk}_i, \mathit{\Pi}_2(1^{\mathbf{k}}, i, x_i, \{\mathit{\Pi}.\mathsf{msg}_{1,j\to i}, \mathit{\Pi}.\mathsf{msg}_{1,i\to j}\}_{j\in[n]}, .; r_i))$
  where $\mathit{\Pi}_2(1^{\mathbf{k}}, i, x_i, \{\mathit{\Pi}.\mathsf{msg}_{1,j\to i}\}_{j\in[n]}, .; r_i)$ is the second round next message function of party P_i in $\mathit{\Pi}$ that takes the messages $\{\mathit{\Pi}.\mathsf{msg}_{1,j\to B}\}_{j\in[n]}$ that were broadcast in the first round as input.
- For each $j \in [n] \setminus \{i\}$, it sends $(\mathsf{GC}_i, \{\{K^{X_\ell,i}_{j,\ell}\}_{j\in[n]}\}_{\ell\in[L]})$ to party P_j over a private channel. Here $X = \mathit{\Pi}.\mathsf{msg}_{1,1\to B} || \ldots || \mathit{\Pi}.\mathsf{msg}_{1,n\to B}$ and X_ℓ denotes the ℓ^{th} bit of X.

Reconstruction. Each party does the following.

- It reconstructs the input wire keys received in the previous round. For each $i \in [n]$ and $\ell \in [L]$, it computes the following. $K^{X_\ell}_{j,\ell} \coloneqq \bigoplus_{i\in[n]} K^{X_\ell,i}_{j,\ell}$ and for each $j \in [n]$, it sets $\mathbf{K}_j[\mathit{\Pi}.\mathsf{msg}_{1,1\to B} || \ldots || \mathit{\Pi}.\mathsf{msg}_{1,n\to B}] \coloneqq (K^{X_1}_{j,1}, \ldots, K^{X_L}_{j,1})$

- For each $j \in [n]$ it evaluates the garbled circuit received in the previous round. $\Pi.\mathsf{msg}_{2,j\to B} := \mathsf{Eval}(\mathsf{GC}_j, \mathbf{K}_j[\Pi.\mathsf{msg}_{1,1\to B}|| \dots ||\Pi.\mathsf{msg}_{1,n\to B}])$
- It runs the reconstruction algorithm of Π on $\{\Pi.\mathsf{msg}_{2,j\to B}\}_{j\in[n]}$ to compute the output.

We defer the security of this protocol to the full-version of our paper [2].

Acknowledgments. The last three authors were supported in part by a DARPA/ARL Safeware Grant W911NF-15-C-0213, and a subaward from NSF CNS-1414023.

References

1. Ananth, P., Choudhuri, A.R., Goel, A., Jain, A.: Round-optimal secure multiparty computation with honest majority. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018, Part II. Lecture Notes in Computer Science, vol. 10992, pp. 395–424. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2018)
2. Ananth, P., Choudhuri, A.R., Goel, A., Jain, A.: Two round information-theoretic MPC with malicious security. IACR Cryptology ePrint Archive 2018, 1078 (2018), https://eprint.iacr.org/2018/1078
3. Applebaum, B., Brakerski, Z., Tsabary, R.: Perfect secure computation in two rounds. In: Theory of Cryptography - 16th International Conference, TCC 2018 (2018), https://eprint.iacr.org/2018/894
4. Applebaum, B., Brakerski, Z., Tsabary, R.: Degree 2 is complete for the round-complexity of malicious mpc (2019), https://eprint.iacr.org/2019/200
5. Badrinarayanan, S., Goyal, V., Jain, A., Kalai, Y.T., Khurana, D., Sahai, A.: Promise zero knowledge and its applications to round optimal MPC. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018, Part II. Lecture Notes in Computer Science, vol. 10992, pp. 459–487. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2018)
6. Bar-Ilan, J., Beaver, D.: Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In: Rudnicki, P. (ed.) 8th ACM Symposium Annual on Principles of Distributed Computing. pp. 201–209. Association for Computing Machinery, Edmonton, Alberta, Canada (Aug 14–16, 1989)
7. Beaver, D.: Multiparty protocols tolerating half faulty processors. In: Brassard, G. (ed.) Advances in Cryptology – CRYPTO'89. Lecture Notes in Computer Science, vol. 435, pp. 560–572. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 1990)
8. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: 22nd Annual ACM Symposium on Theory of Computing. pp. 503–513. ACM Press, Baltimore, MD, USA (May 14–16, 1990)
9. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: 20th Annual ACM Symposium on Theory of Computing. pp. 1–10. ACM Press, Chicago, IL, USA (May 2–4, 1988)
10. Benhamouda, F., Lin, H.: k-round mpc from k-round ot via garbled interactive circuits. Tech. rep. (2018)

11. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd Annual Symposium on Foundations of Computer Science. pp. 136–145. IEEE Computer Society Press, Las Vegas, NV, USA (Oct 14–17, 2001)
12. Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universally composable two-party computation without set-up assumptions. In: Biham, E. (ed.) Advances in Cryptology – EUROCRYPT 2003. Lecture Notes in Computer Science, vol. 2656, pp. 68–86. Springer, Heidelberg, Germany, Warsaw, Poland (May 4–8, 2003)
13. Chaum, D.: The spymasters double-agent problem: Multiparty computations secure unconditionally from minorities and cryptographically from majorities. In: Brassard, G. (ed.) Advances in Cryptology – CRYPTO'89. Lecture Notes in Computer Science, vol. 435, pp. 591–602. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 1990)
14. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: 20th Annual ACM Symposium on Theory of Computing. pp. 11–19. ACM Press, Chicago, IL, USA (May 2–4, 1988)
15. Cramer, R., Damgård, I.: Secure distributed linear algebra in a constant number of rounds. In: Kilian, J. (ed.) Advances in Cryptology – CRYPTO 2001. Lecture Notes in Computer Science, vol. 2139, pp. 119–136. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2001)
16. Damgård, I., Ishai, Y.: Constant-round multiparty computation using a black-box pseudorandom generator. In: Shoup, V. (ed.) Advances in Cryptology – CRYPTO 2005. Lecture Notes in Computer Science, vol. 3621, pp. 378–394. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2005)
17. Feige, U., Kilian, J., Naor, M.: A minimal model for secure computation (extended abstract). In: 26th Annual ACM Symposium on Theory of Computing. pp. 554–563. ACM Press, Montréal, Québec, Canada (May 23–25, 1994)
18. Fischer, M.J., Lynch, N.A.: A lower bound for the time to assure interactive consistency. Inf. Process. Lett. 14(4), 183–186 (1982), https://doi.org/10.1016/0020-0190(82)90033-3
19. Garg, S., Ishai, Y., Srinivasan, A.: Two-round mpc: Information-theoretic and black-box. In: Theory of Cryptography - 16th International Conference, TCC 2018 (2018), https://eprint.iacr.org/2018/909
20. Garg, S., Miao, P., Srinivasan, A.: Two-round multiparty secure computation minimizing public key operations. In: Annual International Cryptology Conference. pp. 273–301. Springer (2018)
21. Garg, S., Srinivasan, A.: Garbled protocols and two-round mpc from bilinear maps. In: Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on. pp. 588–599. IEEE (2017)
22. Garg, S., Srinivasan, A.: Two-round multiparty secure computation from minimal assumptions. In: Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II. pp. 468–499 (2018), https://doi.org/10.1007/978-3-319-78375-8_16
23. Gennaro, R., Ishai, Y., Kushilevitz, E., Rabin, T.: The round complexity of verifiable secret sharing and secure multicast. In: 33rd Annual ACM Symposium on Theory of Computing. pp. 580–589. ACM Press, Crete, Greece (Jul 6–8, 2001)
24. Gennaro, R., Ishai, Y., Kushilevitz, E., Rabin, T.: On 2-round secure multiparty computation. In: Yung, M. (ed.) Advances in Cryptology – CRYPTO 2002. Lecture Notes in Computer Science, vol. 2442, pp. 178–193. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2002)

25. Goldreich, O.: The Foundations of Cryptography - Volume 2, Basic Applications. Cambridge University Press (2004)
26. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A. (ed.) 19th Annual ACM Symposium on Theory of Computing. pp. 218–229. ACM Press, New York City, NY, USA (May 25–27, 1987)
27. Goldwasser, S., Lindell, Y.: Secure multi-party computation without agreement. Journal of Cryptology 18(3), 247–287 (Jul 2005)
28. Halevi, S., Lindell, Y., Pinkas, B.: Secure computation on the web: Computing without simultaneous interaction. In: Rogaway, P. (ed.) Advances in Cryptology – CRYPTO 2011. Lecture Notes in Computer Science, vol. 6841, pp. 132–150. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2011)
29. Ishai, Y., Kumaresan, R., Kushilevitz, E., Paskin-Cherniavsky, A.: Secure computation with minimal interaction, revisited. In: Gennaro, R., Robshaw, M.J.B. (eds.) Advances in Cryptology – CRYPTO 2015, Part II. Lecture Notes in Computer Science, vol. 9216, pp. 359–378. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2015)
30. Ishai, Y., Kushilevitz, E.: Private simultaneous messages protocols with applications. In: Fifth Israel Symposium on Theory of Computing and Systems, ISTCS 1997, Ramat-Gan, Israel, June 17-19, 1997, Proceedings. pp. 174–184 (1997), https://doi.org/10.1109/ISTCS.1997.595170
31. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In: 41st Annual Symposium on Foundations of Computer Science. pp. 294–304. IEEE Computer Society Press, Redondo Beach, CA, USA (Nov 12–14, 2000)
32. Ishai, Y., Kushilevitz, E.: Perfect constant-round secure computation via perfect randomizing polynomials. In: Widmayer, P., Ruiz, F.T., Bueno, R.M., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002: 29th International Colloquium on Automata, Languages and Programming. Lecture Notes in Computer Science, vol. 2380, pp. 244–256. Springer, Heidelberg, Germany, Malaga, Spain (Jul 8–13, 2002)
33. Ishai, Y., Kushilevitz, E.: On the hardness of information-theoretic multiparty computation. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology – EUROCRYPT 2004. Lecture Notes in Computer Science, vol. 3027, pp. 439–455. Springer, Heidelberg, Germany, Interlaken, Switzerland (May 2–6, 2004)
34. Ishai, Y., Kushilevitz, E., Paskin, A.: Secure multiparty computation with minimal interaction. In: Rabin, T. (ed.) Advances in Cryptology – CRYPTO 2010. Lecture Notes in Computer Science, vol. 6223, pp. 577–594. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 2010)
35. Patra, A., Ravi, D.: On the exact round complexity of secure three-party computation. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018, Part II. Lecture Notes in Computer Science, vol. 10992, pp. 425–458. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2018)
36. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In: 21st Annual ACM Symposium on Theory of Computing. pp. 73–85. ACM Press, Seattle, WA, USA (May 15–17, 1989)
37. Yao, A.C.C.: How to generate and exchange secrets. In: Foundations of Computer Science, 1986., 27th Annual Symposium on. pp. 162–167. IEEE (1986)