Two Round Information-Theoretic MPC with Malicious Security

Prabhanjan Ananth* MIT
Arka Rai Choudhuri† JHU
Aarushi Goel‡ JHU
Abhishek Jain§ JHU

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.

*prabhanjan@csail.mit.edu
†achoud@cs.jhu.edu
‡aarushig@cs.jhu.edu
§abhishek@cs.jhu.edu

Contents

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) [Yao86, GMW87, BOGW88, CCD88] 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 [BOGW88, CCD88] 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 [Can01] without relying on external trust [CKL03].

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 [HLP11]. 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., [BIB89, IK00]), the only known two round IT-MPC protocols are due to [IK00, IKP10, IKKP15] 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 [ABT18, GIS18].

1.1 Our Results

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

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. [Gol04]) 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 [GIKR02].

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

Theorem 1.1. *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.*

Impossibility of IT-MPC with Public Reconstruction of Output. All recent two round MPC protocols [ACGJ18, GS18a, GIS18, ABT18, ABT19] satisfy a *public reconstruction of output* property, where given (only) the public transcript of a protocol, any (possibly external) party can compute the output of

the protocol. Unlike these protocols, our protocol in Theorem 1.1 does not satisfy this property. In fact, we show (in Section 6) that any IT-MPC protocol that achieves security with abort and public reconstruction of output property[1] is impossible.

Theorem 1.2. *There does not exist an IT-MPC protocol that achieves security with abort and public reconstruction of output property.*

Protocols over P2P Channels. Our protocol in Theorem 1.1 necessarily uses both broadcast and private point-to-point (P2P) channels for achieving security against malicious adversaries.[2] We next investigate whether it is possible to construct two round IT-MPC against malicious adversaries by using *only* P2P channels.[3]

Our second positive 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 [GL05] is a weakening of the standard notion of security of (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 [FL82, PR18]. This establishes security with selective abort as the strongest achievable standard notion of security in two rounds.

Theorem 1.3. *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 1.1 and 1.3 fully resolve the round complexity of maliciously secure IT-MPC (for NC^1 functions).

Comparison with [ABT18, GIS18, ABT19]. Very recently, Applebaum et al. [ABT18] constructed two round perfectly secure MPC for NC^1 against any $t < n/2$ semi-honest corruptions. Garg et al. [GIS18] 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. [ABT19] 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 consider security with (unanimous) abort in the information-theoretic setting.

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 [GS17, BL18, GS18b] 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: [GS17] use bilinear maps, [GS18b] use two-round oblivious transfer, [BL18, GMS18] use two-round oblivious oblivious transfer and additionally garbled circuits and finally, [ACGJ18, ABT18, GIS18] use information-theoretic MPC protocols. Of particular interest to us is the work of Ananth et al. [ACGJ18] 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.

[1] The exact formulation of this property is slightly more technical; we refer the reader to Section 6.

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

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

Background on [ACGJ18]. 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 [IKP10]). While these protocols do not achieve full-fledged malicious security, they achieve a weaker property termed as *privacy with knowledge of outputs* and [ACGJ18] 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 [GS18b]) 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 [ACGJ18] 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 [ACGJ18] 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 [ACGJ18] 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 [IK02]. 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 [ACGJ18]: 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 [ACGJ18] are no longer compatible with our notion of generalized conforming protocols; recall that since the helper protocols in [ACGJ18] 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 [ACGJ18], 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[4] 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 [IK00]. 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.

[4]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.

- 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 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 [BL18].

- 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 in the broadcast setting.

- 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 3.2.2. This is followed by a construction of this notion in Section 3.2.3.
- In Section 4, we define the notion of generalized conforming protocols. We state the CLC property in Definition 12. A construction of generalized conforming protocol satisfying CLC property is presented in Section 4.1.
- Finally, we present the main construction in Section 5.

It turns out that the protocol realized using the above template satisfies a weaker notion of security called privacy with knowledge of outputs [IKP10]; in this notion, the ideal world simulator is given the flexibility to direct the ideal world functionality to send a value, chosen by the simulator, to the honest parties as the output of the functionality. We present a generic transformation that converts a protocol satisfying privacy with knowledge of outputs into one that satisfies security with abort. This transformation uses a novel tool that we introduce, called *one-time multi-key message authentication codes* (one-time multi-key MACs).

Recall that in a traditional one-time message authentication code, there is a single verifier who can check the authenticity of the MAC signature using the private MAC key. In the multi-key MAC setting, there are many verifiers. Each verifier has its own MAC key. We want the following guarantee: suppose a signer signs a message using the MAC keys of all the verifiers; then each of the verifiers can individually perform the verification check on the signature using their own MAC key. In particular, the verification can be performed without any communication between the verifiers. We postpone discussion on the security property until later.

We now describe our generic transformation: let F be an n-party functionality and suppose our goal is to construct an n-party protocol Π for F satisfying statistical security with abort property. Π proceeds by executing an inner protocol Π' that only satisfies statistical privacy with knowledge of outputs property. The input to the functionality F' associated with Π' consists of the private inputs of all parties $(x_1, \ldots, x_n)$ and the one-time multi-key MAC keys of all the parties. The output of F' is $y = F(x_1, \ldots, x_n)$ along with a signature on this value computed using all the keys. After Π' finishes execution, every party gets y' along with a signature. Every party verifies the signature using their own MAC key; if the signature doesn't verify, they abort, otherwise they output y'. This concludes the description of Π. Note that the round complexity of Π' is the same as the round complexity of Π.

To argue that Π satisfies security with abort, at first it would seem that it just suffices to argue that the adversary cannot output a signature on a value y' different from the implicit output y that verifies w.r.t. to all the keys. Unfortunately, we need a stronger property than this for the following reason: the adversary could still force a signature on y such that some honest parties accept this but the rest of the parties deem

this to be an invalid signature. Recall that all the parties have different MAC keys. If this happens then this would violate security with abort property. We define the notion of multi-key unforgeability appropriately to take into account this issue. We conclude by presenting an information-theoretically secure construction of one-time multi-key message authentication codes in Section 3.3.

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 7.

1.3 Related Work

Since the initial feasibility results [Yao86, GMW87, BOGW88, CCD88], 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 [BGJ+18] for a survey of related works in the dishonest majority setting.

Information-Theoretic MPC. The seminal works of [BOGW88, CCD88] provided the first constructions of polynomial-round IT-MPC protocols for general functionalities. These results were further improved upon in [Bea90, RBO89, Cha90] w.r.t. malicious corruption threshold.

Bar-Ilan and Beaver [BIB89] initiated the study of constant-round IT-MPC protocols. Subsequently, further improvements were obtained by [FKN94, IK97, CD01]. The work of [IK00] 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 [GIKR01]. [IK02] constructed constant round perfectly secure protocols, improving upon the work of [BIB89]. More recently, two round IT-MPC protocols that achieve security with selective abort against $t < n/3$ malicious corruptions were constructed by [IKP10] and [IKKP15]. In fact, [IKP10] and [IKKP15], 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 [FL82, PR18]. All of these positive results are for NC^1 functions; [IK04] established the difficulty of constructing constant-round IT-MPC protocols for general functionalities.

We also highlight the work of [GL05] 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. [BMR90] who constructed such protocols for general functionalities based on one-way functions. Damgård and Ishai [DI05] 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 [IKP10] and [IKKP15] based on one-way functions. Very recently, Ananth et al. [ACGJ18] 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. [ABT18] and Garg et al. [GIS18] also achieve similar results, albeit only against semi-honest adversaries.

2 Preliminaries

We denote the statistical security parameter by $\mathbf{k}$.

2.1 Information-Theoretic MPC

Consider a multiparty computation protocol Π for n parties $P_1, \ldots, P_n$ associated with an n-party functionality $F : \{0,1\}^{\ell_1} \times \cdots \times \{0,1\}^{\ell_n} \to \{0,1\}^{\ell'_1} \times \cdots \times \{0,1\}^{\ell'_n}$, where ℓ_i and ℓ'_i are the input and output lengths of the parties respectively.

We are interested in the following security notions.

Malicious Security. Consider a n-party protocol Π. Let $P_1, \ldots, P_n$ be the set of parties participating in the protocol. To define the malicious security (also referred to as active security in the literature) of Π, we first define the ideal process and the real process below.

Ideal Process: This process is defined with respect to a trusted party. A subset of parties can be corrupted by a PPT ideal process adversary Sim. The process proceeds in the following steps:

1. **Input Distribution**: The environment distributes the inputs $x_1, \ldots, x_n$ to parties $P_1, \ldots, P_n$ respectively.

2. **Inputs to Trusted Party**: The parties now send their inputs to the trusted party. The honest parties send the same input, it received from the environment, to the trusted party. The adversary, however, can send a different input to the trusted party.

3. **Aborting Adversaries**: An adversarial party can then send a message to the trusted party to abort the execution, at this point the trusted party terminates the execution of ideal process. Otherwise, the following steps are executed.

4. **Trusted party answers party** P_i: Suppose the trusted party receives inputs $x'_1, \ldots, x'_n$ from $P_1, \ldots, P_n$ respectively. It sends the i^{th} output y_i, where $F(x'_1, \ldots, x'_n) = (y_1, \ldots, y_n)$ to P_i.

5. **Output**: If the honest party P_i is honest, then it outputs y_i. The adversarial party Sim outputs its entire view.

We denote the adversary participating in the above protocol to be Sim. We define $\mathsf{Ideal}^{\mathsf{Sim},F}(x_1,\ldots,x_n)$ to be the joint distribution defined over the views of the adversary and the outputs of the honest parties.

Security with Selective Abort. 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 aborting adversary can instruct the trusted party to send the output to some honest parties and abort to others. Note that this is slightly different from the standard notion of security with abort, where the aborting adversary can instruct the trusted party to either send abort to all the honest parties or send the output to all the honest parties.

Real Process: Fix a set of inputs $(x_1,\ldots,x_n)$, where $x_i \in \{0,1\}^{\ell_i}$. Party P_i receives the input x_i. All the parties then execute the protocol Π. A subset of parties S is controlled by an adversary $\mathcal{A}$. As in the ideal process, they receive inputs from the environment. $\mathcal{A}$ can deviate arbitrarily from the rules of the protocol. We call such adversaries *malicious* adversaries.

We define $\mathsf{Real}^{\mathcal{A},F}(x_1,\ldots,x_n)$ to be the joint distribution over the view of the adversary and the outputs of the honest parties.

Definition 1 (Statistical/Perfect Malicious Security)**.** *Consider a n-party functionality F as defined above. Fix a set of inputs $(x_1,\ldots,x_n)$, where $x_i \in \{0,1\}^{\ell_i}$. Let Π be a n-party protocol implementing F. We say that Π is* **ε-statistically secure against malicious adversaries** *if for every probabilistic adversary $\mathcal{A}$ controlling a subset of parties S in the real process, there exists a PPT adversary Sim in the ideal process such that:*

$$\mathsf{Ideal}^{\mathsf{Sim},F}(x_1,\ldots,x_n) \approx_{s,\varepsilon} \mathsf{Real}^{\mathcal{A},F}(x_1,\ldots,x_n)$$

If the above two distributions are identical, then we say that Π satisfies **perfect security against malicious adversaries**.

Privacy with Knowledge of Outputs Property. Consider an n-party functionality $F : \{0,1\}^{\ell_1} \times \cdots \times \{0,1\}^{\ell_n} \to \{0,1\}^{\ell'_1} \times \cdots \times \{0,1\}^{\ell'_n}$. To define privacy with knowledge of outputs property, we first define the ideal and real world processes defined with respect to an n-party secure computation protocol for F.

- Ideal Processs, $\mathsf{Ideal}^{\mathsf{Sim},F}(x_1,\ldots,x_n)$: The i^{th} party receives as input x_i from the environment. Every party sends x'_i to the ideal functionality. If a party is honest then $x'_i = x_i$. The parties, controlled by the simulator Sim, receive $F(x_1,\ldots,x_n)$. The simulator then directs the ideal functionality to either deliver the actual output $F(x_1,\ldots,x_n)$ to the honest parties or send them values of his choice. The output of this process comprises of outputs of the honest parties and the view of Sim.
- Real Process, $\mathsf{Real}^{\mathcal{A},F}(x_1,\ldots,x_n)$: The i^{th} party receives as input x_i from the environment. Execute the protocol Π. The output of this process comprises of outputs of the honest parties and the view of $\mathcal{A}$.

Definition 2 (Statistical/Perfect Privacy with Knowledge of Outputs)**.** *An n-party computation protocol securely computing an n-party functionality is said to satisfy perfect privacy with knowledge of outputs property of the following holds: for every function $\mathcal{A}$ corrupting set S of parties with $|S| < \frac{n}{2}$, there exists a PPT simulator Sim such that the following holds:*

$$\left\{\mathsf{Real}^{\mathcal{A},F}(x_1,\ldots,x_n)\right\} \approx_{s,\varepsilon} \left\{\mathsf{Ideal}^{\mathsf{Sim},F}(x_1,\ldots,x_n)\right\},$$

where $\mathsf{Real}^{\mathcal{A},F}$ and $\mathsf{Ideal}^{\mathsf{Sim},F}$ are defined above.

If the above distributions are identically distributed then we say that Π satisfies **perfect privacy with knowledge of outputs** *property.*

Remark. For our applications we only consider protocols with privacy with knowledge of outputs property for $n-$party functionalities of the form $F : \{0,1\}^{\ell_1} \times \ldots \times \{0,1\}^{\ell_n} \to \{0,1\}^{\ell_{\text{out}}}$ where each party has the same output. Note that in this case, according to the above security definition of privacy with knowledge of outputs, the adversary can force different outputs on different honest parties. We consider a slightly stronger variant of this security notion where the correctness of output for the honest parties is still not guaranteed but the adversary cannot force different outputs on different honest parties. Henceforth (unless stated otherwise), we use the term "privacy with knowledge of outputs" to refer to this stronger security notion.

Delayed Function Property. In the context of two-round secure MPC protocols, we define delayed function property to be one where the functionality, that the parties intend to securely compute, is available to them only after the first round.

2.2 Garbling Schemes

A garbling scheme [Yao86] consists of the following algorithms:

- **Setup,** $\mathsf{Gen}(1^{\mathbf{k}}, 1^L, 1^d)$: On input statistical security parameter $\mathbf{k}$, number of leaves L of the circuit to be garbled, depth d of the circuit, it outputs the garbling key gk and input wire labels $\mathbf{K}$.
  Note: We parse $\mathbf{K}$ *as* $(K_1^0, K_1^1, \ldots, K_L^0, K_L^1)$. *For some input* $x \in \{0,1\}^\ell$, $\mathbf{K}[x]$ *denotes* $(K_1^{x_{\xi(1)}}, \ldots, K_L^{x_{\xi(L)}})$, *where* $\xi : [\ell] \to [L]$ *is such that* $x_{\xi(i)}$ *is assigned to the* i^{th} *leaf of the formula.*
- **Garbling,** $\mathsf{Garb}(\mathsf{gk}, C)$: On input garbling key gk, circuit C, output the garbled circuit GC.
- **Evaluation,** $\mathsf{Eval}(\mathsf{GC}, \mathbf{K}[x])$: On input garbled circuit GC, input wire labels $\mathbf{K}[x]$ corresponding to x, output the value y. This is a deterministic algorithm.

We require the following properties from a garbling scheme.

Correctness. Consider a circuit $C : \{0,1\}^\ell \to \{0,1\}^{\ell'}$ of L leaves and depth d. We require that $\mathsf{Eval}(\mathsf{GC}, \mathbf{K}[x]) = C(x)$, where: (i) $(\mathsf{gk}, \mathbf{K}) \leftarrow \mathsf{Gen}(1^{\mathbf{k}}, 1^L, 1^d)$ and, (ii) $\mathsf{GC} \leftarrow \mathsf{Garb}(\mathsf{gk}, C)$.

Security. We require that a garbling of C and the input wire labels corresponding to x don't reveal any information beyond $C(x)$. We formalize this below.

Definition 3. *Let* $\boldsymbol{k}$ *be the statistical security parameter. A garbling scheme* $(\mathsf{Gen}, \mathsf{Garb}, \mathsf{Eval})$ *is said to be* $\varepsilon(\boldsymbol{k})$**-statistical secure** *if for every circuit* C, *input* x, *adversary* $\mathcal{A}$, *there exists a PPT simulator* Sim *with oracle access to* $\mathcal{A}$ *such that the following holds:*

$$\{(\mathsf{GC}, \mathbf{K}[x])\} \approx_{s,\varepsilon(\boldsymbol{k})} \left\{\mathsf{Sim}\left(1^{\boldsymbol{k}}, \varphi(C), C(x)\right)\right\},$$

where (i) $(\mathsf{gk}, \mathbf{K}) \leftarrow \mathsf{Gen}(1^{\boldsymbol{k}}, 1^L, 1^d)$ *and, (ii)* $\mathsf{GC} \leftarrow \mathsf{Garb}(\mathsf{gk}, C)$. *Also,* $\varphi(C)$ *denotes the topology of* C.

If the above two distributions are identically distributed then we say that $(\mathsf{Gen}, \mathsf{Garb}, \mathsf{Eval})$ *satisfies* **perfect security**.

Known Construction(s). Ishai and Kushilevitz [IK02] showed the existence of information-theoretically secure garbling schemes for circuits and in particular for NC^1 circuits, their result yields an efficient (polynomial in circuit size) construction of garbling schemes. We state their result formally below.

Lemma 2.1. *[IK02] Consider a depth-d size-s circuit* C *that takes as input* ℓ *bits and outputs* ℓ' *bits.*

Fix a statistical security parameter $\boldsymbol{k} > 0$. *There is a perfectly secure garbling scheme* $(\mathsf{Gen}, \mathsf{Garb}, \mathsf{Eval})$ *for* C *such that the following two properties hold:*

- *Every input wire label is of length at least* $\boldsymbol{k}$.
- $\mathsf{Gen}, \mathsf{Garb}$ *and* Eval *can each be represented by a* $O(d)$*-depth and* $(\boldsymbol{k} \cdot s)^c \cdot 2^{c \cdot d}$*-sized circuit, for some constant* $c > 0$.

Remark 1. *Note that the complexity of* Gen *upper bounds the size of the input wire labels to be* $(\boldsymbol{k} \cdot s)^c \cdot 2^{c \cdot d}$*, for some constant* c*.*

Remark 2. *We note that the requirement that the length of the input wire labels to be at least* $\boldsymbol{k}$ *is required for our main construction in Section 5. Specifically, we show that our construction is* ε*-statistically secure, where* ε *is negligible in the length of the input wire labels of the garbled circuit and thus, negligible in* $\boldsymbol{k}$*. However, the setting of statistically security parameter is irrelevant for our helper protocols, in Section 3, which will be proved perfectly secure.*

3 Helper Primitives

We consider the following helper primitives 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.
- Finally, we define a new primitive called one-time multi-key MACs and give an information theoretic construction for the same.

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. [IKP10]. Formally, they prove the following lemma.

Lemma 3.1 ([IKP10])**.** *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 statistical 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*.*

Proof. We now sketch a proof for Lemma 3.1. The two-round protocol given by Ishai et al. in [IKP10] makes use of a primitive called multiparty conditional disclosure of secrets (MCDS). The following definition is taken verbatim from [IKP10]:

Definition 4. *An MCDS (multiparty CDS) protocol is a protocol amongst* n *parties, which include three distinct special parties* S, A, B*. The sender* S *holds a secret* s*, and parties* A, B *hold inputs* a, b *(respectively). The protocol should satisfy the following properties:*

- *If* $a = b$*, and* A, B, S *are honest, then all honest parties output* s*.*
- *If* $a = b$*, and* A, B *are honest, then the adversary's view is independent of* a*, even conditioned on* s*.*
- *If* $a \neq b$*, and* A, B, S *are honest, then the adversary's view is independent of* s*, even conditioned on* a, b*.*

As usual, the adversary is allowed to be rushing.

Ishai et al. [IKP10] give a two-round protocol for this multiparty primitive where only three parties participate in the protocol (only party S sends a message in the first round), while all n parties may learn the output. The next message functions of this primitive can be represented by $O(1)$-depth and $O(1)$-sized circuits. We are now ready to give a high level overview of their two-round protocol that achieves privacy with knowledge of outputs. Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the parties in the protocol and $\{x_1, \ldots, x_n\}$ be their respective inputs. The protocol uses a pairwise-verifiable 2 multiplicative (n, t) linear secret sharing scheme (LSSS) over $\mathbb{F}_2$ to share inputs of the parties. Construction of such schemes with the appropriate definitions can be found in [IKP10]. We describe the protocol for a single output bit.

Round 1 Party P_i proceeds as follows:

- Uses a pairwise-verifiable 2 multiplicative (n, t) linear secret sharing scheme (LSSS) over $\mathbb{F}_2$ to compute shares $(s_1^i, \ldots, s_n^i)$ of its input x_i and distributes them among other parties. It also computes and distributes additive shares $(z_1^i, \ldots, z_n^i)$ of 0. This computation can be done using a circuit of size $(n)^c$ and depth $O(\log(n))$.
- Each triple of distinct parties $i, j, k \in [n]$ not holding an input x_h for each $h \in [n]$, participate in the MCDS protocol playing the roles of S, A, B respectively. P_i computes the first round messages for MCDS, where its input is an independent randomly chosen mask $s_{i,j,k,h}$. Since it participates in $\binom{n-1}{3}$ parallel executions of MCDS, these computations can be done using a circuit of size n^c and depth $O(1)$.

Overall the next message function of this round can be implemented by a circuit of depth $O(\log(n))$ and size $(n)^c$.

Round 2 Party P_i proceeds as follows:

- It computes $y_i = G(s_i^1, \ldots, s_i^n) + \sum_{j-1}^{n} z_i^j + \sum_{j,k,h} s_{i,j,k,h}$ and sends y_i to all other parties. This computation can be done using a circuit of size $(n)^c$ and depth $O(\log(n))$.
- It computes second round messages for the $\binom{n-1}{3}$ parallel executions of MCDS. Here a, b are the outputs of the relevant local computations applied to shares of x_h held by P_j, P_k that should ideally be equal. These computations can be done using a circuit of size n^c and depth $O(1)$.

Overall the next message function of this round can also be implemented by a circuit of depth $O(\log(n))$ and size $(n)^c$.

To compute every output bit, we can run multiple executions of this protocol in parallel. This does not affect the overall depth of the circuit implementing the next message function. However, the overall size of the next message function now become $(\ell_{out} \cdot n)^c$. □

Remark. The protocol of [IKP10] 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. See Definition 2 for a detailed discussion.

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.

3.2.1 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 5 (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 6 (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 $G(x_1, f(x_2))$.*

3.2.2 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.

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

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 7 (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 $\{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2} \times \bot \cdots \times \bot$ 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 statistically indistinguishable.*

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 [5]. 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 $\mathsf{Ideal}^{\mathsf{Sim},G}$ as in Section 2.1 – in particular, as in the definition of privacy with knowledge of outputs property in Section 2.1, 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 8 (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 $\{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2} \times \bot \cdots \times \bot$ 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 statistically indistinguishable.*

Remark 3. *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 (Definition 2). Formally, we can analogously define the real world process $\mathsf{Real}^{\mathcal{A},F}$ and ideal world process $\mathsf{Ideal}^{\mathsf{Sim},F}$ as in Section 2.1. We define the security property below.

Definition 9 (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 $\{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2} \times \bot \cdots \times \bot$ 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 10. *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 7), secure against $\mathcal{S}_2$ (Definition 8) and secure against $\mathcal{S}_3$ (Definition 9).*

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

3.2.3 Construction

We prove the following lemma.

Lemma 3.2. *Let $n, \ell_1, \ell_2, \ell' > 0$. Consider a two-input n-party functionality $G : \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2} \times \bot \times \cdots \times \bot \to \{(y, \ldots, y)\}_{y \in \{0,1\}^{\ell'}}$ 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 6) satisfying statistical 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 statistical secure garbling scheme and a secure MPC protocol for quadratic polynomials in the honest majority setting satisfying privacy with knowledge of outputs property (Lemma 3.1). We denote the garbling scheme by $(\mathsf{Gen}, \mathsf{Garb}, \mathsf{Eval})$. We denote the secure MPC protocol for quadratic polynomials by Π_{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 [BL18]. 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:

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 Π_{Quad}. In Π_{Quad}, its input is $\mathbf{K}_I^1$. It sends the first round messages of Π_{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 Π_{Quad}. It also generates a random string R (we define its length below). In Π_{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 Π_{Quad} to other parties.

- P_i, for $i \neq 1, i \neq 2$, generates the first round messages of Π_{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 Π_{Quad}. The protocol Π_{Quad} is associated with a function that takes as input $(\mathbf{K}_I^1, \mathbf{K}_I^2, \bot, \ldots, \bot)$ and outputs $\mathbf{K}_I^1\left[\mathbf{K}_I^2[f] \circ R\right]$[6]. We note that this function can be implemented by a system 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 $(\mathbf{K}_I^2[f], R)$, computes $y \leftarrow \mathsf{Eval}(\mathsf{GC}_2, \mathbf{K}_I^2[f])$ 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 Π_{Quad} along with GC_1.

- P_2 generates the second round messages of Π_{Quad} and sends them to other parties.

- P_i, for $i \neq 1$ and $i \neq 2$, computes the second round messages of Π_{Quad} and sends them to other parties.

[6]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.

Reconstruction. All the parties compute the output of Π_{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.

Correctness. By the correctness of Π_{Quad}, the output of Π_{Quad} is $\mathbf{K} = \mathbf{K}_I^1[\mathbf{K}_I^2[f] \circ R]$; that is the output of Π_{Quad} is the input wire labels for GC_1 corresponding to the input $\mathbf{K}_I^2[f]$. By the correctness of the garbling scheme, we get the output of $\mathsf{Eval}(\mathsf{GC}_2, \mathbf{K}_I^2[f])$ to be $f(x_2)$. Invoking the correctness property of the garbling scheme once more we get the output of $\mathsf{Eval}\left(\mathsf{GC}_1, \mathbf{K}_I^1[\mathbf{K}_I^2[f] \circ R]\right)$ to be $G(x_1, f(x_2))$.

Efficiency. We first determine the garbling and evaluation complexities of GC_1 and GC_2.

From Lemma 2.1, the algorithms $\mathsf{Gen}(1^{\mathbf{k}}, 1^{L''}, 1^{d''})$, $\mathsf{Garb}(\mathsf{gk}_2, U_{x_2})$ and $\mathsf{Eval}(\mathsf{GC}_2, \mathbf{K}_I^2[x_2])$ can each be represented by a $O(d)$-depth $s^{c_1}2^{c_1 d}$-sized circuit, for some constant c_1. Thus, $\widehat{G}$ can be represented by a $O(d)$-depth $s^{c_2}2^{c_2 d}$-sized circuit, for some constant c_2. From Lemma 2.1, the algorithms $\mathsf{Gen}(1^{\mathbf{k}}, 1^{L'}, 1^{d'})$, $\mathsf{Garb}(\mathsf{gk}_1, \widehat{G})$ and $\mathsf{Eval}(\mathsf{GC}_1, \mathbf{K}_I^1[\mathbf{K}_I^2[x_2] \circ R])$ can each be represented by a $O(d)$-depth $s^{c_3}2^{c_3 d}$-sized circuit, for some constant c_3.

We calculate the computational complexities of all the parties in the above protocol.

- Computational complexity of P_1: to calculate this, it suffices to determine the complexity to garble GC_2, the complexity of evaluating GC_2 and the next message function complexity of Π_{Quad}. We have already determined the first two. Moreover, from Lemma 3.1, the next message function of Π_{Quad} can be represented by a $O(\log(n))$-depth n^{c_4}-sized circuit, for some constant c_4. Thus, the next message of P_1 can be represented by a depth-$O(d + \log(n))$ $s^c 2^{c \cdot d}$-sized circuit, for some constant c.

- Computational complexity of P_2: to calculate this, it suffices to determine the complexity to garble GC_1, the complexity of evaluating GC_2 and the next message function complexity of Π_{Quad}. We have already determined the first two. As in the above bullet, the next message function of Π_{Quad} can be represented by a $O(\log(n))$-depth n^{c_4}-sized circuit, for some constant c_4. Thus, the next message of P_2 can be represented by a depth-$O(d + \log(n))$ $s^c 2^{c \cdot d}$-sized circuit, for some constant c.

- Computational complexity of the rest of the parties: to calculate this, it suffices to determine the complexity of evaluating GC_2 and the next message function complexity of Π_{Quad}. We omit this analysis since this follows from the analysis of the above two bullets. As before, we can represent the next message function of P_i, for $3 \leqslant i \leqslant n$, by a depth-$O(d + \log(n))$ $s^c 2^{c \cdot d}$-sized circuit, for some constant c.

3.2.4 Proof of Security

We know prove security of our construction. 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 Π_{Quad} to obtain the first round messages of Π_{Quad}. Generate $R \xleftarrow{\$} \{0,1\}^{|\mathsf{GC}_2|}$. Send the first round messages of Π_{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 Π_{Quad} to obtain the first round messages of Π_{Quad}. Send the first round messages, intended for the parties in S, to the adversary.

  Also, extract the input $\mathbf{K}_I^1$ of P_1 in Π_{Quad} from the first round messages of Π_{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 Π_{Quad}[7], 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.

– 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.

We argue that the output distributions of the ideal world and the real world are identical in this case. We state the hybrids below.

$\mathbf{H}_1$: This corresponds to Expt_0 (Definition 7).

$\mathbf{H}_2$: In this hybrid, the protocol Π_{Quad} is simulated. We define the following hybrid simulator that proceeds as follows.

- **Round 1**.

  – Simulating on behalf of P_2: Execute the simulator of Π_{Quad} to obtain the first round messages of Π_{Quad}. It generates $\mathsf{Gen}(1^{\mathbf{k}}, 1^{L''}, 1^{d''})$ to obtain $(\mathsf{gk}_2, \mathbf{K}_I^2)$. It generates $\mathsf{Garb}(\mathsf{gk}_2, U_{x_2})$ to obtain GC_2. Generate $R \xleftarrow{\$} \{0,1\}^{|\mathsf{GC}_2|}$. Send the first round messages of Π_{Quad} intended for the parties in S along with $\mathsf{GC}_2 \oplus R$ to the adversary.

  – Simulating on behalf of parties in $\overline{S}\backslash\{P_2\}$: Execute the simulator of Π_{Quad} to obtain the first round messages of Π_{Quad}. Send the first round messages intended for the parties in S to the adversary.

  Also, extract the input $\mathbf{K}_I^1$ of P_1 in Π_{Quad} from the first round messages of Π_{Quad}.

- **Round 2**.

  – Simulating on behalf of P_2: Execute the simulator of Π_{Quad}, with the output of Π_{Quad} set to be $\mathbf{K}_I^1\left[\widehat{\mathbf{K}_I^2}[f] \circ R\right]$, 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.

  – Simulating on behalf of parties in $\overline{S}\backslash\{P_2\}$: 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. 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.

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

In particular, for every party not in the set S, the simulator generates the simulated messages of Π_{Quad}.

From the statistical privacy with knowledge of outputs property of Π_{Quad}, the output distributions of $\mathbf{H}_1$ and $\mathbf{H}_2$ are statistically indistinguishable.

$\mathbf{H}_3$: This corresponds to Expt_1 (Definition 7).

The only difference between $\mathbf{H}_2$ and $\mathbf{H}_3$ is that the garbled circuit GC_2 (computed by P_2) in $\mathbf{H}_3$ is simulated whereas the garbled circuit in $\mathbf{H}_2$ is honestly generated. Moreover, exactly one wire label of GC_2 per input wire is used to simulate the messages of Π_{Quad}. Thus we can invoke the statistical security of $(\mathsf{Gen}, \mathsf{Garb}, \mathsf{Eval})$ and argue that the output distributions of $\mathbf{H}_2$ and $\mathbf{H}_3$ are statistically indistinguishable.

$P_2 \in S$ and $P_1 \notin S$. 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.

Consider the following hybrids.

$\mathbf{H}_1$: Execution of the protocol in the real world.

$\mathbf{H}_2$: In this hybrid, the messages of Π_{Quad} are simulated. We define a hybrid simulator that proceeds 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_2.

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 $\widehat{y}$. Send $\widehat{y}$ to the ideal functionality to receive $\widehat{b}$.

  – Simulating on behalf of P_1: Compute $\mathsf{Gen}(1^{\mathbf{k}}, 1^{L'}, 1^{d'})$ to obtain $(\mathsf{gk}_\cdot, \mathbf{K}_I^1)$. Compute $\mathsf{Garb}(\mathsf{gk}_1, \widehat{G})$ to obtain GC_1. Execute the simulator of Π_{Quad}, with the output of Π_{Quad} set to be $\widehat{\mathbf{K}_I^1}\left[\widehat{\mathbf{K}_I^2} \circ R\right]$, to generate the second round messages of Π_{Quad}. Send the second round messages of Π_{Quad} intended for the parties in S along with the garbled circuit GC_1 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 $\widehat{\mathbf{K}_I^1}\left[\widehat{\mathbf{K}_I^2} \circ R\right]$, 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. 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.

The only difference between $\mathbf{H}_1$ and $\mathbf{H}_2$ is that the messages of Π_{Quad} are simulated in $\mathbf{H}_2$, whereas in $\mathbf{H}_1$ they are not. From the statistical privacy with knowledge of outputs property of Π_{Quad}, it follows that the output distributions of $\mathbf{H}_1$ and $\mathbf{H}_2$ are statistically indistinguishable.

$\mathbf{H}_3$: This corresponds to the ideal world.

The only difference between $\mathbf{H}_2$ and $\mathbf{H}_3$ is that the garbled circuit GC_1 is simulated in $\mathbf{H}_3$, whereas in $\mathbf{H}_2$ it is generated honestly. Moreover, exactly one wire label of GC_1 per input wire is used in Π_{Quad}. Thus, from the statistical security of $(\mathsf{Gen}, \mathsf{Garb}, \mathsf{Eval})$ it follows that the output distributions of $\mathbf{H}_2$ and $\mathbf{H}_3$ are statistically indistinguishable.

$P_1 \notin S$ 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 $\widehat{\mathbf{K}_I^1}$ 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.

We describe the hybrids below.

$\mathbf{H}_1$: Execution of the protocol in the real world.

$\mathbf{H}_2$: In this hybrid, the messages of Π_{Quad} are simulated. We define a hybrid simulator that proceeds 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 of Π_{Quad}. Compute $\mathsf{Gen}(1^{\mathbf{k}}, 1^{L''}, 1^{d''})$ to obtain $(\mathsf{gk}_2, \mathbf{K}_I^2)$. Generate a random string $R \xleftarrow{\$} \{0,1\}^{|\mathsf{GC}_1|}$. Compute $\mathsf{Garb}(\mathsf{gk}_2, U_{x_2})$ to obtain GC_2.

    Finally, it sends $\mathsf{GC}_2 \oplus R$ along with the first round simulated messages of Π_{Quad}, intended for 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_1.

- **Round 2**. After the first round, the simulator receives the function f from the environment.

  – Simulating on behalf of P_1: Compute $\mathsf{Gen}(1^{\mathbf{k}}, 1^{L'}, 1^{d'})$ to obtain $(\mathsf{gk}_1, \mathbf{K}_I^1)$. Compute $\mathsf{Garb}(\mathsf{gk}_1, \widehat{G})$ to obtain GC_1. Simulate the second round messages of Π_{Quad}, with the output of Π_{Quad} set to be $\mathbf{K}_I^1\left[\mathbf{K}_I^2[f]\right]$. Send the simulated second round messages of Π_{Quad}, intended for parties in S, to the adversary.

  – Simulating on behalf of P_2: Simulate the second round messages of Π_{Quad}, with the output of Π_{Quad} set to be $\mathbf{K}_I^1\left[\mathbf{K}_I^2[f]\right]$. Send the simulated second round messages of Π_{Quad}, intended for 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 $\widehat{\mathbf{K}_I^1}$ 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.

The only difference between $\mathbf{H}_1$ and $\mathbf{H}_2$ is that the messages of Π_{Quad} are simulated. From the statistical security of Π_{Quad}, it follows that the output distributions of $\mathbf{H}_1$ and $\mathbf{H}_2$ are statistically indistinguishable.

$\mathbf{H}_3$: In this hybrid, the hybrid simulator, defined in $\mathbf{H}_2$, instead simulates the garbled circuit GC_1. In particular, the hybrid simulator generates GC_1 as follows: it executes the simulator $\mathsf{Sim}_{GC}(\varphi(U_{x_2}), \widehat{y})$ to obtain $\left(\mathsf{GC}_1, \mathbf{K}_I^2[f]\right)$, where $\widehat{y} = f(x_2)$. The rest of the steps of the hybrid simulator are same as before.

The only difference between $\mathbf{H}_2$ and $\mathbf{H}_3$ is that the garbled circuit GC_1 is simulated in $\mathbf{H}_3$, whereas in $\mathbf{H}_2$ it is honestly generated. From the statistical security of $(\mathsf{Gen}, \mathsf{Garb}, \mathsf{Eval})$, the output distributions of $\mathbf{H}_2$ and $\mathbf{H}_3$ are statistically indistinguishable. Here we crucially use the fact that only one wire label (of GC_1) per input wire is necessary to generate the messages of the protocol.

$\mathbf{H}_4$: This corresponds to the ideal world.

The only difference between $\mathbf{H}_3$ and $\mathbf{H}_4$ is that the masked garbled circuit sent by P_2 in the first round is generated in $\mathbf{H}_3$ as $R \oplus \mathsf{GC}_2$, whereas in $\mathbf{H}_3$ just R is sent as the first message. Note that R is not used in generating the other messages; this means that $R \oplus \mathsf{GC}_2$ and R are identically distributed even given the other messages of the protocol. Thus, the output distributions of $\mathbf{H}_3$ and $\mathbf{H}_4$ are identically distributed.

□

3.3 Information-Theoretic One-Time Multi-Key MACs

In this section we first formally define this new-primitive called one-time multi-key message authentication codes (MACs) and then give a construction for the same. As the name suggests, a multi-key MAC scheme is a multi-key variant of regular unconditional MACs. Each key in this scheme can be sampled independently. Computing a MAC with respect to a set of keys requires knowledge of all the keys. However, verification can be done individually w.r.t. each key.

Intuitively for correctness, we want that an honestly computed MAC can be verified w.r.t. each key. For multi-key unforgeability, we allow the adversary to choose keys $\{K_i\}_{i \in S}$ for any subset $S \subset [n]$ and receive a valid MAC τ corresponding to the set of $\{K_i\}_{i \in [n]}$ keys (where the remaining keys $\{K_i\}_{i \in [n] \setminus S}$ corresponding to the honest set are sampled independently and honestly) on a message x of its choice. We say that a multi-key MAC scheme is multi-key unforgeable if the probability that the adversary can output a different MAC $\tau^* \neq \tau$ such that it verifies w.r.t. any key K_i in the honest set, i.e., for $i \in [n] \setminus S$ is negligible. We now define this notion formally.

Definition 11. *A multi-key one-time MAC scheme with message space $\mathcal{M}$, key-space $\mathcal{K}$ and signature space $\mathcal{X}$ for a set of parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ is a tuple of 3 PPT algorithms $\Phi := (\mathsf{MK.KeyGen}, \mathsf{MK.Sign}, \mathsf{MK.Verify})$.*

- $K \leftarrow \mathsf{MK.KeyGen}(1^{\boldsymbol{k}})$*: The key generation algorithm takes the statistical security parameter as input and outputs a key $K \in \mathcal{K}$.*
- $\tau \leftarrow \mathsf{MK.Sign}(K_1, \ldots, K_n, x)$*: The signing algorithm takes a set of keys $K_1, \ldots, K_n \in \mathcal{K}^n$, and a message bit $x \in \mathcal{M}$ as input and outputs a MAC $\tau \in \mathcal{X}$.*
- $b \leftarrow \mathsf{MK.Verify}(K, x, \tau)$*: The verification algorithm takes a key $K \in \mathcal{K}$, a message bit $x \in \mathcal{M}$ and a MAC value $\tau \in \mathcal{X}$ as input and outputs a bit $b \in \{0, 1\}$.*

We say that a multi-key one-time MAC scheme is $\varepsilon(\boldsymbol{k})$-statistically secure if it satisfies the following properties:

1. ***Correctness:*** *The following holds for every $x \in \mathcal{M}$:*

$$\Pr\left[\{K_i \leftarrow \mathsf{MK.KeyGen}(1^{\boldsymbol{k}})\}_{i \in [n]}, \tau \leftarrow \mathsf{MK.Sign}(K_1, \ldots, K_n, x) \mid \forall i \in [n], \mathsf{MK.Verify}(K_i, x, \tau) = 1\right] = 1$$

2. ***Multi-Key Unforgeability:*** *For any unbounded adversary $\mathcal{A}$, the following holds:*

$$\Pr\left[\mathsf{Expt}^{\mathsf{forge}}_{\Phi,\mathcal{A}}(1^{\boldsymbol{k}}, 1^n) = 1\right] \leqslant \varepsilon(\boldsymbol{k})$$

where $\mathsf{Expt}^{\mathsf{forge}}_{\Phi,\mathcal{A}}$ is defined as follows:

- *The Adversary $\mathcal{A}$ chooses a subset $S \subset [n]$, a set of keys $\{K_i\}_{i\in[n]\setminus S}$ and a message $x \in \mathcal{M}$ and sends these to the challenger.*
- *The challenger samples keys $\{K_i \leftarrow \mathsf{MK.KeyGen}(1^{\boldsymbol{k}})\}_{i\in S}$ and computes $\tau \leftarrow \mathsf{MK.Sign}(K_1, \ldots, K_n, x)$ such that for each $i \in [n]$, $\mathsf{MK.Verify}(K_i, x, \tau) = 1$. It sends τ to the adversary.*
- *The adversary $\mathcal{A}$ returns x^*, τ^*.*
- *Output 1 if $\tau \neq \tau^*$ and $\exists i \in S$ such that $\mathsf{MK.Verify}(K_i, x^*, \tau^*) = 1$. Else output 0.*

We now construct a statistically secure multi-key MAC scheme for message space $\mathcal{M} = \{0, 1\}$. The keys in this scheme comprise of three elements $A \in \mathbb{F}^n, b_0 \in \mathbb{F}, b_1 \in \mathbb{F}$, where $\mathbb{F}$ is a field of size $2^{\mathbf{k}}$. Computing a MAC on a message $x \in \{0, 1\}$ w.r.t. n such keys of the form $\{(K_i, b_{i,0}, b_{i,1})\}_{i\in[n]}$ is equivalent to finding a vector $U = \tau$ of size n, such that $\langle A_i, U\rangle = b_{i,x}$ for each $i \in [n]$. Verification w.r.t. to any key $K_i = (A_i, b_{i,0}, b_{i,1})$ can also be done similarly by checking whether $\langle A_i, U\rangle = b_{i,x}$. We now formally prove the following lemma.

Lemma 3.3. *Given a statistical security parameter $\boldsymbol{k} > 0$, there exists an $\varepsilon(\boldsymbol{k})$-statistically secure one-time multi-key MAC scheme $\Phi := (\mathsf{MK.KeyGen}, \mathsf{MK.Sign}, \mathsf{MK.Verify})$ for message space $\mathcal{M} = \{0, 1\}$.*

Proof. We start by describing the construction. Later we give a proof of security for this construction.
Construction. Let $\mathbb{F}$ be a field of size exponential in the statistical security parameter $\mathbf{k}$.

- $\mathsf{MK.KeyGen}(1^{\mathbf{k}})$: Sample $A \xleftarrow{\$} \mathbb{F}^n$ and $b_0, b_1 \xleftarrow{\$} \mathbb{F}$. Output $K = (A, b_0, b_1)$.
- $\mathsf{MK.Sign}(K_1, \ldots, K_n, x)$: $\forall i \in [n]$, parse $K_i = (A_i, b_{i,0}, b_{i,1})$. Let $A = \begin{bmatrix} A_1 \\ \cdot \\ \cdot \\ A_n \end{bmatrix}$ and $B_x = \begin{bmatrix} b_{1,x} \\ \cdot \\ \cdot \\ b_{n,x} \end{bmatrix}$. Sample a vector U such that $A \cdot U = B_x$. Output $\tau = U$.
- $\mathsf{MK.Verify}(K_i, x, \tau)$: Parse $K_i = (A_i, b_{i,0}, b_{i,1})$ and $\tau = U$. Check whether $\langle A_i, U\rangle = b_{i,x}$. If so output 1, else output 0.

Correctness. Correctness follows trivially from construction. Since U is chosen such that $A \cdot U = B_x$, then it holds for each $i \in [n]$ that $\langle A_i, U\rangle = b_{i,x}$.

Multi-Key Unforgeability. Given any $i \in S$, Let $A_i = [a_{i,1}, \ldots, a_{i,n}]$ and $\tau = U$ and $\tau^* = U^*$. Since $b_{i,\overline{x}}$ is chosen uniformly at random, the probability that there exists $U^{*T} = [u_1^* \ldots u_n^*]$ such that $b_{i,\overline{x}} = a_{i,1} \cdot u_1^* + \ldots + a_{i,n} \cdot u_n^*$ is $1/|\mathbb{F}|$. Therefore,

$$\Pr[\mathsf{MK.Verify}(K_i, \overline{x}, \tau^*) = 1] \leqslant 1/|\mathbb{F}|$$

For $\mathsf{MK.Verify}(K_i, x, \tau^*) = 1$, since $\tau \neq \tau^*$, U and U^* must differ in at least one position. Let $u_j^* \neq u_j$. Then

$$a_{i,j} = \frac{b_{i,x} - \sum_{k\in[n]\setminus j} a_{i,k} \cdot u_k^*}{u_j^*}$$

Since $a_{i,j}$ is chosen uniformly at random, the probability that this happens is $1/|\mathbb{F}|$.

$$\Pr[\mathsf{MK.Verify}(K_i, x, \tau^*) = 1] \leqslant 1/|\mathbb{F}|$$

Therefore, $\Pr[\mathsf{MK.Verify}(K_i, x^*, \tau^*) = 1] \leqslant 2/|\mathbb{F}|$ for any $x^* \in \{0, 1\}$.

Finally, taking a union bound:

$$\Pr\left[(\exists i \in S \text{ such that } \mathsf{MK.Verify}(K_i, \cdot, \tau^*) = 1) \wedge (\tau \neq \tau^*)\right] \leqslant 2|S|/|\mathbb{F}|$$

□

Note that the above construction can be easily extended to obtain a one-time multi-key MAC scheme for multi-bit message space $\mathcal{M} = \{0,1\}^*$. This can be done by computing a multi-key MAC for each bit of the message separately using the construction from Lemma 3.3. As a result, we get the following corollary.

Corollary 3.4. *Given a statistical security parameter $\boldsymbol{k} > 0$, there exists an $\varepsilon(\boldsymbol{k})$-statistically secure one-time multi-key MAC scheme $\Phi := (\mathsf{MK.KeyGen}, \mathsf{MK.Sign}, \mathsf{MK.Verify})$ for a multi-bit message space $\mathcal{M} = \{0,1\}^*$.*

4 Generalized Conforming Protocols

The notion of conforming protocols was first defined in [GS18b] 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:

$$\mathbf{st}_i := \left(R_k : \forall k \in S_i \bigcup_{i\neq i'} T_{i,i'} \bigcup_{i\neq i'} T_{i',i} \right)$$

  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 $\mathbf{Z}^{i,1}_k = 0$. Here $\mathbf{Z}^{i,1}_k$ 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 $\mathbf{Z}^{i,1}_k$ 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 $(\mathbf{L}^I_{i,j}, \mathbf{C}_{i,j}, \mathbf{L}^O_{i,j})$.
- If $j \neq 1$, for $\{(k, z_k)\}_{\forall i' \neq i,\ k \in \mathbf{L}^O_{i',j-1}}$, 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 $\mathbf{L}^I_{i,j}$ along with $\mathbf{st}_i$, compute $\mathbf{C}_{i,j}$ and update the locations in $\mathbf{Z}^j$ specified by the set $\mathbf{L}^O_{i,j}$. Call the resulting state $\mathbf{Z}^{i,j+1}$.
- Send all the updated values and locations $\{(k, z_k)\}_{k \in \mathbf{L}^O_{i,j}}$ 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 12 (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 $k \in \mathbf{L}^O_{i,1}$.

4.1 Construction

We prove the following lemma.

Lemma 4.1. *Let $n, \ell_1, \ell'_1, \ldots, \ell_n, \ell'_n > 0$. Consider an n-party functionality $F : \{0,1\}^{\ell_1} \times \cdots \times \{0,1\}^{\ell_n} \to \{0,1\}^{\ell'_1} \times \cdots \times \{0,1\}^{\ell'_n}$ 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.*

Proof. We start with any deterministic[8] MPC protocol Π, that is information theoretically secure against malicious adversaries and runs in a constant number of rounds t. Let $(t+1)^{th}$ round correspond to the output computation phase. Also, the next message functions in Π must be implementable by a circuit of depth $O(d+\log(s))$ and size $s^c 2^{c\cdot(d+\log(s))}$. The protocols constructed by [BIB89, IK00] satisfy this property.

We assume without loss of generality that each next message function in Π takes input of maximum length ℓ_{in} and computes an output of maximum length ℓ_{out}. Since, each party is allowed to send a broadcast message as well as private messages to every party in each round, we further assume that the output of each next message function is divided into $n+1$ messages of equal length. Each of these conditions can be met by suitably padding the messages with 0s.

Since such a protocol requires the use of private channels, we add an extra pre-processing phase where the parties exchange one-time pads (of suitable length) to emulate the private channels over a broadcast channel in subsequent rounds. In particular, whenever a party P_i has to send a message to another party P_j , it encrypts its message using the one-time pad P_i had chosen and sent to P_j in the pre-processing phase. Now we transform such an interactive MPC protocol into a conforming protocol as follows:

$|S_i| = |x_i| + (t+1)\cdot(\ell_{in}+\ell_{out})$, where x_i is the input of party P_i. $|T_{i_1,i_2}| = t\cdot(\frac{\ell_{out}}{n+1})$ and $|U| = t\cdot n\cdot(\frac{\ell_{out}}{n+1})$. Each of these partitions are arranged in the global state as follows:

$$\mathbf{Z} := S_1||T_{1,2}||\dots||T_{1,n}||\dots||S_i||T_{i,1}||\dots||T_{i,n}||\dots||T_{n,n-1}||U$$

. Let $x_1,\dots,x_n$ denote the respective inputs of each party.

- **Pre-processing Phase.** For each $i\in[n]$, party P_i defines $\mathbf{st}_i$ to be the list:

$$\mathbf{st}_i := \left(R_k : \forall k \in S_i \bigcup_{i\neq i'} T_{i,i'} \bigcup_{i\neq i'} T_{i',i}\right)$$

  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 $\mathbf{Z}^{i,1}_k = 0$. Here $\mathbf{Z}^{i,1}_k$ denotes the k^{th} bit of $\mathbf{Z}^{i,1}$
  - Let $S_i[h]$ denote the h^{th} location in S_i and $S_i[0:h]$ denote the first h locations in set S_i. Set $z_{S_i[0:|x_i|]} = x_i \oplus (R_{S_i[0]}||\dots||R_{S_i[|x_i-1|]})$. Compute

$$\mathsf{Pre}(1^{\mathbf{k}}, i, x_i, \mathbf{st}_i) = \{(z_k, k) : k\in S_i[0:|x_i|]\}$$

    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 $\mathbf{Z}^{i,1}_k$ 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

$$\mathbf{Z}^1 = \bigoplus_{i-1}^{n}\mathbf{Z}^{i,1}$$

  It proceeds as follows:

[8] Randomized protocols can be viewed as deterministic protocols where the randomness used by a party is a part of its input.

- Parse the action $\Phi_{i,1}$ as $(\mathbf{L}^I_{i,1}, \mathbf{C}_{i,1}, \mathbf{L}^O_{i,1})$.
- $\mathbf{L}^I_{i,1} = S_i[0 : |x_i|]$
- Compute $x_i = z_{S_i[0:|x_i|]} \oplus (R_{S_i[0]}|| \dots ||R_{S_i[|x_i-1|]})$ using $\mathbf{st}_i$ and $\mathbf{Z}^1$
- Compute $\Pi_{i,1} \leftarrow \mathsf{NMF}_{i,1}(1^{\mathbf{k}}, x_i)$, where $\mathsf{NMF}_{i,1}(.) = \widehat{C}_{i,1}$ is the first round next message function of P_i. Let $\Pi_{i,1} = \Pi^{i\to 1}_{i,1}|| \dots ||\Pi^{i\to n}_{i,1}||\Pi^B_{i,1}$.
- It encrypts $\Pi^{i\to i}_{i,1}$, using the appropriate random bits from $\mathbf{st}_i$ and updates the appropriate locations $\mathbf{L}^O_{i,1} \subset S_i$ in Z^1 with these encrypted values. Note that these will be next $\frac{\ell_{out}}{n+1}$ locations is S_i after $|x_i|$.
- For $i' \neq i$, it encrypts $\Pi^{i\to i'}_{i,1}$, using the appropriate random bits from $\mathbf{st}_i$ and updates the appropriate locations $\mathbf{L}^O_{i,1} \subset T_{i,i'}$ in Z^1 with these encrypted values. Note that these will be the first $\frac{\ell_{out}}{n+1}$ locations in $T_{i,i'}$.
- It also updates the next $\frac{\ell_{out}}{n+1}$ vacant locations in U specified by $\mathbf{L}^O_{i,1}$ with $\Pi^B_{i,1}$.
- Call the resulting state $\mathbf{Z}^{i,j+1}$ and send all the updated values and locations $\{(k, z_k)\}_{k\in\mathbf{L}^O_{i,j}}$ to all other parties.

For each $j \in [t+1] \setminus \{1\}$, party P_i proceeds as follows:

- Parse $\Phi_{i,j}$ as $(\mathbf{L}^I_{i,j} = \mathbf{L}^{I\to}_{i,j} \cup \mathbf{L}^{I\leftarrow}_{i,j},\ \mathbf{C}_{i,j},\ \mathbf{L}^O_{i,j} = \mathbf{L}^{O\to}_{i,j} \cup \mathbf{L}^{O\leftarrow}_{i,j})$.
- $\mathbf{L}^{I\to}_{i,j}$ consists of all those location in $U \bigcup_{i'\in[n]\setminus\{i\}} T_{i,i'} \bigcup_{i'\in[n]\setminus\{i\}} T_{i',i}$ where the masked input to $\mathsf{NMF}_{i,j}$ is stored.
- **Copy Operation.** Each party P_i firsts updates the k^{th} location in $\mathbf{Z}^{i,j}[\mathsf{copy1}]$ with the value z_k where $\{(k, z_k)\}_{\forall i'\neq i, k\in\mathbf{L}^{O\leftarrow}_{i',j-1}}$ (if $j = 2$, $\mathbf{L}^{O\leftarrow}_{i',j-1} = \mathbf{L}^O_{i',j-1}$) are the values received from other parties in the previous step. The resulting state is called $\mathbf{Z}^j[\mathsf{copy1}]$. The first copy operation corresponds to unmasking these messages, re-masking them with the appropriate random bits from $\mathbf{st}_i$ and copying them to the locations specified by $\mathbf{L}^{I\leftarrow}_{i,j} \subset S_i$. Each party P_i sends the updated locations values and locations $\{(k, v_k)\}_{k\in\mathbf{L}^{I\leftarrow}_{i,j}}$ to all other parties. Now that all the required inputs to $\mathsf{NMF}_{i,j}$ are stored in the private computation space of P_i, we are ready to proceed with the local operation.
- **Local Operation.** Each party P_i firsts updates the k^{th} location in $\mathbf{Z}^{i,j}[\mathsf{local}]$ with the value z_k where $\{(k, z_k)\}_{\forall i'\neq i, k\in\mathbf{L}^{I\leftarrow}_{i',j}}$ are the values received from other parties in the previous step. The resulting state is called $\mathbf{Z}^j[\mathsf{local}]$. Local computation essentially corresponds to computing $\mathsf{NMF}_{i,j}$ using the unmasked values stored in S_i. This operation takes the appropriate masked values in S_i and $\mathbf{st}_i$ as input. The output of this computation is masked using the appropriate random bits from $\mathbf{st}_i$ and stored in the next ℓ_{out} vacant positions in S_i. Each party P_i sends the updated locations values and locations $\{(k, v_k)\}_{k\in S_i}$ to all other parties.
- **Copy Operation.** Each party P_i firsts updates the k^{th} location in $\mathbf{Z}^{i,j}[\mathsf{copy2}]$ with the value z_k where $\{(k, z_k)\}_{\forall i'\neq i, k\in S_{i'}}$ are the values received from other parties in the previous step. The resulting state is called $\mathbf{Z}^j[\mathsf{copy2}]$. The second copy operation corresponds to unmasking the values specified by $\mathbf{L}^{O\to}_{i,j} \subset S_i$, re-masking them using the appropriate random bits from $\mathbf{st}_i$ and copying them to the locations specified by $\mathbf{L}^{O\leftarrow}_{i,j}$. Each party P_i sends the updated locations values and locations $\{(k, v_k)\}_{k\in\mathbf{L}^{O\leftarrow}_{i,j}}$ to all other parties. Sending the broadcast and private channel messages to other parties in each round is captured by this operation.

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 Phase.** Each party unmasks the last ℓ'_i values in S_i to learn the output.

□

We need to argue that Φ preserves the correctness and security guarantees of Π. Note that Φ is essentially the same as Π with the only difference being that there is an additional pre-processing phase in Φ and all the remaining private channel messages are sent over a broadcast channel. Each round in Π, corresponds to 3 operations/rounds in Φ - one for decrypting private messages sent over a broadcast channel, the second one for computing the next message function and the third one for encrypting private messages before sending them over a broadcast channel. This does not affect the correctness guarantees of Π. Also, each of these private messages are suitably encrypted using masks sampled uniformly at random in the pre-processing phase. This ensures that the security properties of Π also remain intact. We should point out that the way the first copy operation is defined, some bits might get copied multiple times by the same party. However this redundancy does not affect the overall asymptotic complexity of our protocol.

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 5.1. *Let $n, \ell_1, \ldots, \ell_n, \ell_{\mathsf{out}} > 0$. Consider an n-party single-output functionality $F : \{0,1\}^{\ell_1} \times \cdots \times \{0,1\}^{\ell_n} \to \{0,1\}^{\ell_{\mathsf{out}}}$ computable by a depth-d circuit of size s.*

Fix a statistical security parameter $\boldsymbol{k} > 0$. There is a maliciously secure two-round MPC protocol for F with $\mathsf{negl}(\boldsymbol{k})$-statistical security with abort 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.

Looking ahead, we first give a construction for a two-round secure MPC protocol for F that achieves $\mathsf{negl}(\mathbf{k})$-statistical privacy with knowledge of outputs in Section 5.1. Later in Section 5.2, we use a multi-key MAC scheme to transform this protocol into one that achieves $\mathsf{negl}(\mathbf{k})$-statistical security with abort.

5.1 A Two-round Secure MPC Satisfying Privacy with Knowledge of Outputs

We now construct a two-round MPC protocol that achieves statistical privacy with knowledge of outputs.

Lemma 5.2. *Let $n, \ell_1, \ldots, \ell_n, \ell_{\mathsf{out}} > 0$. Consider an n-party single-output functionality $F : \{0,1\}^{\ell_1} \times \cdots \times \{0,1\}^{\ell_n} \to \{0,1\}^{\ell_{\mathsf{out}}}$ computable by a depth-d circuit of size s.*

There is a maliciously secure two-round MPC protocol for F with statistical privacy with knowledge of outputs in the honest majority setting. Moreover, the computational complexity of this protocol is polynomial in s and exponential in d.

5.1.1 Construction.

We begin by describing the tools required for our construction.

Tools. We use the following ingredients in our construction.

- t-round Generalized Conforming protocol for F, guaranteed by Lemma 4.1. 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 3.1.
- Delayed-function two-round secure n-party MPC for specialized two-input functionalities, as guaranteed by Lemma 3.2.
- Information-theoretic garbling scheme $(\mathsf{Gen}, \mathsf{Garb}, \mathsf{Eval})$ from Definition 2.2.

The construction proceeds as follows:

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 1.

- For every $j \in [t+1] \setminus \{1\}$, 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 2.
  - $(\mathsf{gk}_{i,j}[\mathsf{local}], \mathbf{K}_{i,j}[\mathsf{local}]) \leftarrow \mathsf{Gen}(1^{\mathbf{k}}, 1^L, 1^d)$, where L is the number of leaves and d is the depth of the formula in Figure 3.
  - $(\mathsf{gk}_{i,j}[\mathsf{copy2}], \mathbf{K}_{i,j}[\mathsf{copy2}]) \leftarrow \mathsf{Gen}(1^{\mathbf{k}}, 1^L, 1^d)$, where L is the number of leaves and d is the depth of the formula in Figure 4.

- **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 3.1. 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 $\Pi_{\mathsf{Quad}}[i_1, i_2, 1, 1]$ 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 12.
    * 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 $\Pi_{\mathsf{Quad}}[i_1, i_2, 1, 1].\mathsf{msg}_{1, i_4 \to i_5}$ to be the first round message of $\Pi_{\mathsf{Quad}}[i_1, i_2, 1, 1]$ 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 Π_{Quad} instantiations.

  – $\{\Pi_{\mathsf{Quad}}[i_1, i_2, i_3, j, 1]\}_{i_1, i_3 \in [n], i_1 \neq i_3, i_2 \in [n+1] \;\; j \in [t+1] \setminus \{1\}}$
    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 $\Pi_{\mathsf{Quad}}[i_1, i_2, i_3, j, 1]$ 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 $\{\{R_{k'}\}_{k \in T_{i_1,i'}}\}_{i' \in [n] \setminus \{1\}}$.

[9] Recall that $\mathbf{L}^O_{i_1,1}$ 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.

* 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.

– $\{\Pi_{\mathsf{Quad}}[i_1, i_2, j, 2]\}_{i_1,i_2 \in [n], i_1 \neq i_2,\ j \in [t] \setminus \{1\}}$
  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 $\Pi_{\mathsf{Quad}}[i_1, i_2, j, 1]$ 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.

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 $O(n^2 t)$ executions of delayed-function two-round secure n-party MPC, as guaranteed by Lemma 3.2. Denote these instantiations to be $\{\Pi_{\mathsf{DFunc}}[i_1, i_2, j]\}_{i_1,i_2 \in [n], j \in [t+1] \setminus \{1\}}$. 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 $\Pi_{\mathsf{DFunc}}[i_1, i_2, j]$ 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 $\Pi_{\mathsf{DFunc}}[i_1, i_2, j].\mathsf{msg}_{1, i \to i''}$ to be the first message of $\Pi_{\mathsf{DFunc}}[i_1, i_2, j]$ sent by the i^{th} party to $(i'')^{th}$ party.

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 1. 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 2.
  – $(\mathsf{GC}_{i,j}[\mathsf{local}], \mathbf{K}_{i,j}[\mathsf{local}]) \leftarrow \mathsf{Garb}(\mathsf{gk}_{i,j}[\mathsf{local}], C_{i,j}[\mathsf{local}])$, where $C_{i,j}[\mathsf{local}]$ is defined in Figure 3.
  – $(\mathsf{GC}_{i,j}[\mathsf{copy2}], \mathbf{K}_{i,j}[\mathsf{copy2}]) \leftarrow \mathsf{Garb}(\mathsf{gk}_{i,j}[\mathsf{copy2}], C_{i,j}[\mathsf{copy2}])$, where $C_{i,j}[\mathsf{copy2}]$ is defined in Figure 4.
- The i^{th} party broadcasts the following message:

$$\left(\mathsf{GC}_{i,1},\ \mathbf{K}_{i,1}\left[\mathbf{Z}^1\right], \{\mathsf{GC}_{i,j}[\mathsf{copy1}], \mathsf{GC}_{i,j}[\mathsf{local}], \mathsf{GC}_{i,j}[\mathsf{copy2}]\}_{j \in [t+1]}\right)$$

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 $\{\Pi_{\mathsf{Quad}}[i_1, i_2, 1, 1].\mathsf{msg}_{2,i'\to i''}\}_{i_1,i_2,i',i''\in[n],i_1\neq i_2}$. 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 $\{\Pi_{\mathsf{Quad}}[i_1, i_2, i_3, j, 1].\mathsf{msg}_{2,i'\to i''}\}_{i_1,i_3,i',i''\in[n],i_1\neq i_3,i_2\in[n+1]}$.
  - 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 $\{\Pi_{\mathsf{DFunc}}[i_1, i_2, j, 1].\mathsf{msg}_{2,i'\to i''}\}_{i_1,i_2,i',i''\in[n],i_1\neq i_2}$.
  - 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 $\{\Pi_{\mathsf{Quad}}[i_1, i_2 j, 1].\mathsf{msg}_{2,i'\to i''}\}_{i_1,i_2,i',i''\in[n],i_1\neq i_2}$.
  - 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_{out} locations of S_i in $\mathbf{Z}_{fin}$ to compute the output.

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 12), 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 $\mathbf{Z}^{i,2}[\mathsf{copy1}]$.

Figure 1: Description of $C_{i,1}$

Correctness. Recall that at the beginning of each round j of the computation phase in the generalized conforming protocol, each party reconstructs the global invariant state $\mathbf{Z}^j$ before proceeding with its next action $\Phi_{i,j}$. Each $\Phi_{i,j}$, for $j \in [t+1] \setminus 1$ is further subdivided into three operations: copy1, local and copy2. The parties exchange updated information at the end of each of these operations and each party is able to re-construct the following global invariants:

- $\mathbf{Z}^j[\mathsf{copy1}] = \mathbf{Z}^j$ at the beginning of the first copy operation in round j.
- $\mathbf{Z}^j[\mathsf{local}]$ at the beginning of the local operation in round j.
- $\mathbf{Z}^j[\mathsf{copy2}]$ at the beginning of the second copy operation in round j.

In order to ensure correctness of our two-round protocol, we want to ensure that these global invariant states are maintained at every step. Since we are restricted with two-rounds, we implement each of these actions $\Phi_{i,j}$ using a series of garbled circuits that can "talk" to each other. Corresponding to each round $j \in [t+1] \setminus 1$ of the computation phase, each party P_i computes and broadcasts three garbled circuits. Each of these garbled circuits are responsible for performing its assigned operation (copy1, local or copy2) on the global invariant state and provide labels corresponding to this updated state for its next garbled circuit. Information about these updated locations must also be communicated to the next round garbled circuit of all other parties. To enable this we make use of a two-round delayed function MPC for quadratic functionalities from lemma 3.1 and a two-round delayed function MPC for two-input functionalities from lemma 3.2.

We argue correctness using the following claims.

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 12.

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 $\mathbf{Z}^{i,j}[\mathsf{local}]$.

Figure 2: Description of $C_{i,j}[\mathsf{copy1}]$, for $j>1$.

Claim 1. *For each $i\in[n]$, $GC_{i,1}$ has access to the global invariant state $\mathbf{Z}^1$.*

Proof. At the beginning of round 2, all parties compute $\mathbf{Z}^1=\bigoplus_{i=1}^{n}\mathbf{Z}^{i,1}$. As described in Fig 1, the circuit $C_{i,1}$ takes this global invariant state $\mathbf{Z}^1$ as input. Party P_i sends the wire keys $\mathbf{K}_{i,1}[\mathbf{Z}^1]$ corresponding to $\mathbf{Z}^1$ along with garbled circuit $\mathsf{GC}_{i,1}$. Hence $GC_{i,1}$ has access to the global invariant state $\mathbf{Z}^1$. □

Claim 2. *For each $i\in[n]$, $GC_{i,2}[\mathsf{copy1}]$ has access to the global invariant state $\mathbf{Z}^2[\mathsf{copy1}]=\mathbf{Z}^2$.*

Proof. As described in Fig 2, the input to $C_{i,2}[\mathsf{copy1}]$ is of the form $\left(\mathbf{Z}^{i,2}[\mathsf{copy1}],\ \{(k,z_k)\}_{\forall i'\neq i,\ k\in\mathbf{L}^{O}_{i',1}}\right)$. Wire labels corresponding to $\mathbf{Z}^{i,2}[\mathsf{copy1}]$ are output by the garbled circuit $\mathsf{GC}_{i,1}$. For every $i_1\in[n]\setminus\{i\}$, each garbled circuit $\mathsf{GC}_{i',1}$ for $i'\in[n]$, outputs second round messages for $\Pi_{\mathsf{Quad}}[i_1,i,1,1]$ that outputs labels in $\mathbf{K}_{i,2}[\mathsf{copy1}]$ corresponding to $\{(k,z_k)\}$ for every location $k\in\mathbf{L}^{O}_{i_1,1}$ that is updated by $i_1\in[n]\setminus\{i\}$ in the first round of the computation phase in the generalized conforming protocol.

On evaluating $\{\mathsf{GC}_{i',1}\}_{i'\in[n]}$, we can first compute the output for these executions of Π_{Quad} to compute wire labels corresponding to $\{(k,z_k)\}_{\forall i'\neq i,\ k\in\mathbf{L}^{O}_{i',1}}$ and then use these to evaluate garbled circuit $\mathsf{GC}_{i,2}[\mathsf{copy1}]$. $\mathsf{GC}_{i,2}[\mathsf{copy1}]$ first computes $\mathbf{Z}^2=\mathbf{Z}^2[\mathsf{copy1}]$ using input $\left(\mathbf{Z}^{i,2}[\mathsf{copy1}],\ \{(k,z_k)\}_{\forall i'\neq i,\ k\in\mathbf{L}^{O}_{i',1}}\right)$ and then proceeds with the first copy operation for $j=2$. □

Claim 3. *For each $i\in[n]$, $j\in[t+1]\setminus 1$, $GC_{i,j}[\mathsf{copy1}]$ has access to the global invariant state $\mathbf{Z}^j[\mathsf{copy1}]=\mathbf{Z}^j$.*

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 $\mathbf{Z}^{i,j}[\mathsf{copy2}]$.

Figure 3: Description of $C_{i,j}[\mathsf{local}]$, for $j>1$.

Proof. As described in Fig 2, the input to $C_{i,j}[\mathsf{copy1}]$ is of the form $\left(\mathbf{Z}^{i,j}[\mathsf{copy1}],\ \{(k,z_k)\}_{\forall i'\neq i,\ k\in\mathbf{L}^{O\leftarrow}_{i',j-1}}\right)$. Wire labels corresponding to $\mathbf{Z}^{i,j}[\mathsf{copy1}]$ are output by the garbled circuit $\mathsf{GC}_{i,j-1}[\mathsf{copy2}]$. For every $i_1\in[n]\setminus\{i\}$, each garbled circuit $\mathsf{GC}_{i',j-1}[\mathsf{copy2}]$ for $i'\in[n]$, outputs second round messages for $\Pi_{\mathsf{Quad}}[i_1,i,j-1,1]$ that outputs labels in $\mathbf{K}_{i,j}[\mathsf{copy1}]$ corresponding to $\{(k,z_k)\}$ for every location $k\in\mathbf{L}^{O\leftarrow}_{i_1,j-1}$ that is updated by $i_1\in[n]\setminus\{i\}$ in the second copy operation of the $(j-1)^{th}$ round of the computation phase in the generalized conforming protocol.

On evaluating $\{\mathsf{GC}_{i',j-1}[\mathsf{copy2}]\}_{i'\in[n]}$, we can first compute the output for these executions of Π_{Quad} to compute wire labels corresponding to $\{(k,z_k)\}_{\forall i'\neq i,\ k\in\mathbf{L}^{O\leftarrow}_{i',j-1}}$ and then use these to evaluate garbled circuit $\mathsf{GC}_{i,j}[\mathsf{copy1}]$. $\mathsf{GC}_{i,j}[\mathsf{copy1}]$ first computes $\mathbf{Z}^{j}=\mathbf{Z}^{j}[\mathsf{copy1}]$ using input $\left(\mathbf{Z}^{i,j}[\mathsf{copy1}],\ \{(k,z_k)\}_{\forall i'\neq i,\ k\in\mathbf{L}^{O\leftarrow}_{i',j-1}}\right)$ and then proceeds with the first copy operation for round j. □

Claim 4. *For each $i\in[n]$, $j\in[t+1]\setminus 1$, $GC_{i,j}[\mathsf{local}]$ has access to the global invariant state $\mathbf{Z}^{j}[\mathsf{local}]$.*

Proof. As described in Fig 3, the input to $C_{i,j}[\mathsf{local}]$ is of the form $\left(\mathbf{Z}^{i,j}[\mathsf{local}],\ \{(k,z_k)\}_{\forall i'\neq i,\ k\in\mathbf{L}^{I\leftarrow}_{i',j}}\right)$. Wire labels corresponding to $\mathbf{Z}^{i,j}[\mathsf{local}]$ are output by the garbled circuit $\mathsf{GC}_{i,j}[\mathsf{copy1}]$. For every $i_1\in[n]\setminus\{i\}$, $i_2\in[n+1]$ each garbled circuit $\mathsf{GC}_{i',j}[\mathsf{copy1}]$ for $i'\in[n]$, outputs second round messages for $\Pi_{\mathsf{Quad}}[i_1,i_2,i,j,1]$ that output labels in $\mathbf{K}_{i,j}[\mathsf{local}]$ corresponding to $\{(k,z_k)\}$ for every location $k\in\mathbf{L}^{I\rightarrow}_{i_1,j}$ that is updated by $i_1\in[n]\setminus\{i\}$ in the first copy operation of the $(j)^{th}$ round of the computation phase in the generalized conforming protocol.

On evaluating $\{\mathsf{GC}_{i',j}[\mathsf{copy1}]\}_{i'\in[n]}$, we can first compute the output for these executions of Π_{Quad} to compute wire labels corresponding to $\{(k,z_k)\}_{\forall i'\neq i,\ k\in\mathbf{L}^{I\leftarrow}_{i',j}}$ and then use these to evaluate garbled circuit

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 $\left(\mathbf{Z}^{i,j+1}[\mathsf{copy1}], \{(k, z_k)\}_{\forall i' \neq i,\ k \in \mathbf{L}^{O\leftarrow}_{i',j}}\right)$. 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 $\{(k, z_k)\}_{\forall i' \neq i, \mathbf{L}^{O\leftarrow}_{i',j}}$.
- 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 12.

  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 $\mathbf{Z}^{i,j+1}[\mathsf{copy1}]$.

Figure 4: Description of $C_{i,j}[\mathsf{copy2}]$, for $j > 1$.

$\mathsf{GC}_{i,j}[\mathsf{local}]$. $\mathsf{GC}_{i,j}[\mathsf{local}]$ first computes $\mathbf{Z}^j[\mathsf{local}]$ using input $\left(\mathbf{Z}^{i,j}[\mathsf{local}],\ \{(k, z_k)\}_{\forall i' \neq i,\ k \in \mathbf{L}^{I\leftarrow}_{i',j}}\right)$ and then proceeds with the local operation for round j. □

Claim 5. *For each $i \in [n]$, $j \in [t] \setminus \{1\}$, $GC_{i,j}[\mathsf{copy2}]$ has access to the global invariant state $\mathbf{Z}^j[\mathsf{copy2}]$.*

Proof. As described in Fig 4, the input to $C_{i,j}[\mathsf{copy2}]$ is of the form $\left(\mathbf{Z}^{i,j}[\mathsf{copy2}],\ \{(k, z_k)\}_{\forall i' \neq i,\ k \in S_{i'}}\right)$. Wire labels corresponding to $\mathbf{Z}^{i,j}[\mathsf{copy2}]$ are output by the garbled circuit $\mathsf{GC}_{i,j}[\mathsf{local}]$. For every $i_2 \in [n] \setminus \{i\}$, each garbled circuit $\mathsf{GC}_{i',j}[\mathsf{local}]$ for $i' \in [n]$, outputs second round messages for $\Pi_{\mathsf{DFunc}}[i, i_2, j]$ that output labels in $\mathbf{K}_{i,j-1}[\mathsf{copy2}]$ corresponding to $\{(k, z_k)\}$ for every location $k \in S_{i_2}$ that is updated by $i_2 \in [n] \setminus \{i\}$ in the local operation of the $(j)^{th}$ round of the computation phase in the generalized conforming protocol.

On evaluating $\{\mathsf{GC}_{i',j}[\mathsf{local}]\}_{i' \in [n]}$, we can first compute the output for these executions of Π_{Quad} to compute wire labels corresponding to $\{(k, z_k)\}_{\forall i' \neq i,\ k \in S_{i'}}$ and then use these to evaluate garbled circuit $\mathsf{GC}_{i,j}[\mathsf{copy2}]$. $\mathsf{GC}_{i,j}[\mathsf{copy2}]$ first computes $\mathbf{Z}^j[\mathsf{copy2}]$ using input $\left(\mathbf{Z}^{i,j}[\mathsf{copy2}],\ \{(k, z_k)\}_{\forall i' \neq i,\ k \in S_{i'}}\right)$ and then proceeds with the second copy operation for round j. □

Thus, each garbled circuit receives all the relevant information about the values for all the locations that were updated by the previous set of garbled circuits. It also receives information about the locations that were not updated by any party from its own garbled circuit. Together, these are sufficient to reconstruct the global invariant state. Correctness now follows from the correctness of the conforming protocol.

Efficiency analysis. We prove a sequence of claims.

Claim 6. *For every $i \in [n]$, $|C_{i,t+1}[\mathsf{copy2}]| = s^c 2^{c \cdot d}$, for some constant c. Moreover, the depth of $C_{i,t+1}[\mathsf{copy2}]$ is $O(d)$.*

Proof. To prove this, it suffices to upper bound the computational complexity and depth of the next message function of the generalized conforming protocol. By Lemma 4.1, the next message function of the generalized conforming protocol can be implemented by a $O(d)$-depth and $s^{c_1} 2^{c_1 \cdot d}$-sized circuit, for some constant c_1.

Thus, $|C_{i,t+1}[\mathsf{copy2}]| = s^c 2^{c \cdot d}$, for some constant c, and the depth of $C_{i,t+1}[\mathsf{copy2}]$ is $O(d)$. □

Claim 7. *For every $i \in [n], j \in [t]$, $|C_{i,j}[\mathsf{copy2}]| \leqslant (\boldsymbol{k} \cdot s \cdot |C_{i,j+1}[\mathsf{copy1}]|)^c \cdot 2^{c \cdot (d + \log(s))}$, for some constant c. Moreover, the depth of $|C_{i,j+1}[\mathsf{copy1}]|$ is $O(d + \log(s))$.*

Proof. To prove this, it suffices to upper bound the following quantities:

- Complexity of the next message function of the generalized conforming protocol. By Lemma 4.1, the next message function of the generalized conforming protocol can be implemented by a $O(d + \log(s))$-depth and $s^{c_1} 2^{c_1 \cdot (d + \log(s))}$-sized circuit, for some constant c_1.
- Complexity of the next message function of Π_{Quad}. From Lemma 3.1, the next message function of Π_{Quad} can be represented by a $O(\log(n))$-depth $(\ell_{out} \cdot n)^{c_2}$-sized circuit, for some constant c_2, where ℓ_{out} is the number of polynomials associated with Π_{Quad}. From the description of $C_{i,j}[\mathsf{copy2}]$, observe that ℓ_{out} is $s^{c_3} \cdot 2^{c_3 \cdot d}$, for some constant c_3[11].
- Total length of the input wire labels $\mathbf{K}_{i,j+1}[\mathsf{copy1}]$, for every $i \in [n]$. By Lemma 2.1, this quantity is $(\mathbf{k} \cdot |C_{i,j+1}[\mathsf{copy1}]|)^{c_4} \cdot 2^{c_4 \cdot d}$, for some constant c_4.

Thus for every $i \in [n], j \in [t]$, $C_{i,j}[\mathsf{copy2}]$ has depth $O(d + \log(s))$ and size at most $(\mathbf{k} \cdot s \cdot |C_{i,j+1}[\mathsf{copy1}]|)^c \cdot 2^{c \cdot (d + \log(s))}$, for some constant c. □

Claim 8. *For every $i \in [n], j \in [t+1]$, $|C_{i,j}[\mathsf{local}]| \leqslant (\boldsymbol{k} \cdot s \cdot |C_{i,j}[\mathsf{copy2}]|)^c \cdot 2^{c \cdot (d + \log(s))}$, for some constant c. Moreover, the depth of $C_{i,j}[\mathsf{local}]$ is $O(d + \log(s))$.*

Proof. To prove this, it suffices to upper bound the following quantities:

- Computational complexity of the next message function of the generalized conforming protocol. By Lemma 4.1, the next message function of the generalized conforming protocol can be implemented by a $O(d + \log(s))$-depth and $s^{c_1} 2^{c_1 \cdot (d + \log(s))}$-sized circuit, for some constant c_1.
- Computational complexity of the next message function of Π_{DFunc}. By Lemma 3.2, the next message function of Π_{DFunc} can be represented by a $O(d + \log(s))$-depth $s^{c_2} 2^{c_2 \cdot (d + \log(s))}$-sized circuit, for some constant c_2.
- Total length of the input wire labels $\mathbf{K}_{i,j}[\mathsf{copy2}]$. By Lemma 2.1, this quantity is $(\mathbf{k} \cdot |C_{i,j}[\mathsf{copy2}]|)^{c_3} \cdot 2^{c_3 \cdot d}$, for some constant c_3.

Thus for every $i \in [n], j \in [t+1]$, $C_{i,j}[\mathsf{local}]$ has depth $O(d + \log(s))$ size at most $(\mathbf{k} \cdot s \cdot |C_{i,j}[\mathsf{copy2}]|)^c \cdot 2^{c \cdot (d + \log(s))}$, for some constant c. □

Claim 9. *For every $i \in [n], j \in [t+1] \setminus \{1\}$, $|C_{i,j}[\mathsf{copy1}]| \leqslant (\boldsymbol{k} \cdot s \cdot |C_{i,j}[\mathsf{local}]|)^c \cdot 2^{c \cdot (d + \log(s))}$, for some constant c. Moreover, the depth of $C_{i,j}[\mathsf{copy1}]$ is $O(d + \log(s))$.*

Proof. To prove this, it suffices to upper bound the following quantities:

- Computational complexity of the next message function of the generalized conforming protocol. By Lemma 4.1, the next message function of the generalized conforming protocol can be implemented by a $O(d + \log(s))$-depth and $s^{c_1} 2^{c_1 \cdot (d + \log(s))}$-sized circuit, for some constant c_1.

[11] Specifically, ℓ_{out} is upper bounded by the number of locations in the global state $\mathbf{Z}$ modified in any given round; which in turn is upper bounded by the length of the global state $\mathbf{Z}$ in any round.

- Computational complexity of the next message function of Π_{Quad}. From Lemma 3.1, the next message function of Π_{Quad} can be represented by a $O(\log(n))$-depth $(\ell_{out} \cdot n)^{c_2}$-sized circuit, for some constant c_2, where ℓ_{out} is the number of polynomials associated with Π_{Quad}. From the description of $C_{i,j}[\mathsf{copy1}]$, observe that ℓ_{out} is $s^{c_3} \cdot 2^{c_3 \cdot d}$, for some constant c_3.
- Total length of the input wire labels $\mathbf{K}_{i,j}[\mathsf{local}]$. By Lemma 2.1, this quantity is $(\mathbf{k} \cdot |C_{i,j}[\mathsf{local}]|)^{c_4} \cdot 2^{c_4 \cdot d}$, for some constant c_4.

Thus, for every $i \in [n], j \in [t+1]\backslash\{1\}$, $C_{i,j}[\mathsf{copy1}]$ has depth $O(d + \log(s))$ and size $(\mathbf{k} \cdot s \cdot |C_{i,j}[\mathsf{local}]|)^c \cdot 2^{c \cdot (d + \log(s))}$, for some constant c.

□

Claim 10. *For every $i \in [n]$, $|C_{i,1}| \leqslant (\boldsymbol{k} \cdot s \cdot |C_{i,2}[\mathsf{copy1}]|)^c \cdot 2^{c \cdot (d + \log(s))}$, for some constant c.*

Proof. The proof of this claim follows along the same lines as the proof of Claim 9. □

Consider the following claim.

Claim 11. *For every $i \in [n]$, the next message function of i^{th} party can be represented by a circuit of depth $O(d + \log(s))$ and size $n \cdot (\boldsymbol{k} \cdot s)^c \cdot 2^{c \cdot (d + \log(s))}$, for some constant c.*

Proof. Let $i \in [n]$. We first calculate $|C_{i,1}|$. There exists a constant c' such that the following holds,

$$\begin{aligned}
|C_{i,1}| &\leqslant (\mathbf{k} \cdot s \cdot |C_{i,2}[\mathsf{copy1}]|)^{c'} \cdot 2^{c' \cdot (d + \log(s))} \\
&\leqslant \left(\mathbf{k} \cdot s \cdot \left((\mathbf{k} \cdot s \cdot |C_{i,2}[\mathsf{local}]|) \cdot 2^{c' \cdot (d + \log(s))}\right)\right)^{c'} \cdot 2^{c' \cdot (d + \log(s))} \\
&= \left(\mathbf{k}^2 \cdot s^2 \cdot |C_{i,2}[\mathsf{local}]|\right)^{c'} \cdot 2^{(c'^2 + c') \cdot (d + \log(s))} \\
&\leqslant \cdots \\
&\leqslant \left(\mathbf{k}^{3(t+1)} \cdot s^{3(t+1)} \cdot |C_{i,t+1}[\mathsf{copy2}]|\right)^{c'} \cdot 2^{\left(c' + \cdots + c'^{3(t+1)}\right) \cdot (d + \log(s))} \\
&\leqslant \left(\mathbf{k}^{3(t+1)} \cdot s^{3(t+1)} \cdot (\mathbf{k} \cdot s)^{c'} \cdot 2^{c' \cdot (d + \log(s))}\right)^{c'} \cdot 2^{\left(c' + \cdots + c'^{3(t+1)}\right) \cdot (d + \log(s))}
\end{aligned}$$

Since t is a constant, we have that $|C| \leqslant (\mathbf{k} \cdot s)^{c''} \cdot 2^{c'' \cdot d}$, for some constant d. Furthermore, observe that for every $i \in [n], j \in [t+1]\backslash\{1\}$, $|C_{i,1}| \geqslant |C_{i,j}[\mathsf{copy1}]|$. Similarly, for every every $i \in [n], j \in [t+1]$, $|C_{i,1}| \geqslant |C_{i,j}[\mathsf{local}]|$ and $|C_{i,1}| \geqslant |C_{i,j}[\mathsf{copy2}]|$.

We are now ready to prove the claim: since the garbling of all the circuits $C_{i,1}, \{C_{i,j}[\mathsf{copy1}], C_{i,j}[\mathsf{local}], C_{i,j}[\mathsf{copy2}]\}_{j \in [t+1]\backslash\{1\}}$ can be done in parallel, the depth of the circuit representing the next message function of the i^{th} party is governed by the sum of the following quantities: (i) depth of the next message function of Π_{Quad}, (ii) depth of the next message function of Π_{DFunc} and, (iii) maximum depth of the garbling of circuits $C_{i,1}, \{C_{i,j}[\mathsf{copy1}], C_{i,j}[\mathsf{local}], C_{i,j}[\mathsf{copy2}]\}_{j \in [t+1]\backslash\{1\}}$. From Lemma 3.1, Lemma 3.2 and the above claims, the depth of the circuit representing the next message function of the i^{th} party is $O(d + \log(s))$.

We now upper bound the size of the circuit representing the i^{th} next message function. This is governed by the sum of the following quantities: (i) size of the next message function of Π_{Quad}, (ii) size of the next message function of Π_{DFunc} and, (iii) n times the garbling complexity of circuit $C_{i,1}$. From Lemma 3.1, Lemma 3.2, (i) and (ii) can be computed by a circuit of size polynomial in $\mathbf{k}, s$ and exponential in $(d + \log(s))$. Combining this with an upper bound on $|C_{i,1}|$ determined above, we have that the circuit representing the next message function of the i^{th} party has size at most $(\mathbf{k} \cdot s)^c \cdot 2^{c \cdot d}$, for some constant c. □

5.1.2 Proof of Security

Simulator. We now give a description of the Ideal world Simulator $\mathcal{S}$. $\mathcal{S}$ internally uses the simulator Sim_Φ of the underlying conforming protocol Φ, simulator $\mathsf{Sim}_{\mathsf{GC}}$ for the statistically secure garbing scheme GC and

simulators $\mathsf{Sim}_{\Pi_{\mathsf{Quad}}}$ and $\mathsf{Sim}_{\Pi_{\mathsf{DFunc}}}$ of protocols Π_{Quad} and Π_{DFunc} respectively. We can think of the simulator Sim_{GC} to consist of two parts, namely Sim^1_{GC} and Sim^2_{GC}, where the Sim^1_{GC} takes as input the statistical security parameter $\mathbf{k}$ and the topology $\varphi(C)$ of the circuit and outputs simulated garbled key and simulated input wire keys $(\mathsf{gk}, \mathbf{K})$. The second part Sim^2_{GC} takes the output of the first part and the output of the garbled circuit $C(x)$ as input and outputs a simulated garbled circuit GC. Similarly the simulator $\mathsf{Sim}_{\Pi_{\mathsf{Quad}}}$ can also be divided into two parts, $(\mathsf{Sim}^1_{\Pi_{\mathsf{Quad}}}, \mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}})$ that simulate transcripts for the two rounds. The simulator $\mathsf{Sim}_{\Pi_{\mathsf{DFunc}}}$ can also be divided into two parts $(\mathsf{Sim}^1_{\Pi_{\mathsf{DFunc}}}, \mathsf{Sim}^2_{\Pi_{\mathsf{DFunc}}})$ to simulate transcripts for the two rounds separately. For a detailed description of this simulator for the different cases, see section 3.2.3. Let $\mathcal{H}$ be the set of honest parties and $\mathcal{A}$ be the adversary. The simulator $\mathcal{S}$ proceeds as follows:

Round 1: $\mathcal{S} \to \mathcal{A}$:

- The simulator $\mathcal{S}$ internally invokes the simulator Sim_{Φ} to obtain simulated first round (pre-processing phase) messages of Φ.
$$\{\mathbf{Z}^{i,1}, (R_k : \forall k \in \bigcup_{i' \notin \mathcal{H}} T_{i,i'})\}_{i \in \mathcal{H}} \leftarrow \mathsf{Sim}_{\Phi}(1^{\mathbf{k}})$$
It sends this to adversary.

- The simulator $\mathcal{S}$ invokes the simulator $\mathsf{Sim}^1_{\mathsf{GC}}$ to obtain simulated messages,
$$\{(\mathsf{gk}_{i,1}, \mathbf{K}_{i,1})\}_{i \in \mathcal{H}} \leftarrow \{\mathsf{Sim}^1_{\mathsf{GC}}(1^{\mathbf{k}}, \varphi(C_{i,1}))\}_{i \in \mathcal{H}}$$
where $\varphi(C_{i,1})$ is the topology of the circuit in Figure 1.

- For every $j \in [t+1] \backslash \{1\}$ and $i \in \mathcal{H}$, $\mathcal{S}$ computes the following:
$$(\mathsf{gk}_{i,j}[\mathsf{copy1}], \mathbf{K}_{i,j}[\mathsf{copy1}]) \leftarrow \mathsf{Sim}^1_{\mathsf{GC}}(1^{\mathbf{k}}, \varphi(C_{i,j[\mathsf{copy1}]}))$$
where where $\varphi(C_{i,j}[\mathsf{copy1}])$ is the topology of the circuit in Figure 2.
$$(\mathsf{gk}_{i,j}[\mathsf{local}], \mathbf{K}_{i,j}[\mathsf{local}]) \leftarrow \mathsf{Sim}^1_{\mathsf{GC}}(1^{\mathbf{k}}, \varphi(C_{i,j[\mathsf{local}]}))$$
where where $\varphi(C_{i,j}[\mathsf{local}])$ is the topology of the circuit in Figure 3.
$$(\mathsf{gk}_{i,j}[\mathsf{copy2}], \mathbf{K}_{i,j}[\mathsf{copy2}]) \leftarrow \mathsf{Sim}^1_{\mathsf{GC}}(1^{\mathbf{k}}, \varphi(C_{i,j[\mathsf{copy2}]}))$$
where where $\varphi(C_{i,j}[\mathsf{copy2}])$ is the topology of the circuit in Figure 4.

- For every $i_1, i_2, \in [n]$ and $i_1 \neq i_2$, the simulator invokes the simulator $\mathsf{Sim}^1_{\Pi_{\mathsf{Quad}}}$ of Π_{Quad} to obtain simulated first round messages of the honest parties and sends them to the adversary.
$$\{\Pi_{\mathsf{Quad}}[i_1, i_2, 1, 1].\mathsf{msg}_{1, i_4, \to i_5}\}_{i_4 \in \mathcal{H}, i_5 \notin \mathcal{H}} \leftarrow \mathsf{Sim}^1_{\Pi_{\mathsf{Quad}}}(1^{\mathbf{k}})$$

- For every $i_1, i_3 \in [n], i_1 \neq i_3, i_2 \in [n+1], j \in [t+1] \setminus \{1\}$, the simulator invokes the simulator $\mathsf{Sim}_{\Pi_{\mathsf{Quad}}}$ of Π_{Quad} to obtain simulated first round messages of the honest parties and sends them to the adversary.
$$\{\Pi_{\mathsf{Quad}}[i_1, i_2, i_3, j, 1].\mathsf{msg}_{1, i_4, \to i_5}\}_{i_4 \in \mathcal{H}, i_5 \notin \mathcal{H}} \leftarrow \mathsf{Sim}^1_{\Pi_{\mathsf{Quad}}}(1^{\mathbf{k}})$$

- For every $i_1, i_2 \in [n], i_1 \neq i_2, j \in [t+1] \setminus \{1\}$, the simulator invokes the simulator $\mathsf{Sim}_{\Pi_{\mathsf{Quad}}}$ of Π_{Quad} to obtain simulated first round messages of the honest parties and sends them to the adversary.
$$\{\Pi_{\mathsf{Quad}}[i_1, i_2, j, 2].\mathsf{msg}_{1, i_4, \to i_5}\}_{i_4 \in \mathcal{H}, i_5 \notin \mathcal{H}} \leftarrow \mathsf{Sim}^1_{\Pi_{\mathsf{Quad}}}(1^{\mathbf{k}})$$

- For every $i_1, i_2 \in [n], i_1 \neq i_2, j \in [t] \setminus \{1\}$, the simulator invokes the simulator $\mathsf{Sim}_{\Pi_{\mathsf{DFunc}}}$ of Π_{DFunc} to obtain simulated first round messages of the honest parties and sends them to the adversary.
$$\{\Pi_{\mathsf{DFunc}}[i_1, i_2, j].\mathsf{msg}_{1, i' \to i''}\}_{i \in \mathcal{H}, i'' \notin \mathcal{H}} \leftarrow \mathsf{Sim}^1_{\Pi_{\mathsf{DFunc}}}(1^{\mathbf{k}})$$

Round 1: $\mathcal{A} \to \mathcal{S}$: The simulator receives first round messages from $\mathcal{A}$ on behalf of every $i \in \mathcal{A}$.

- Receive $\{\mathbf{Z}^{i,1}, (R_k : \forall k \in \bigcup_{i' \in \mathcal{H}} T_{i,i'})\}_{i \in \mathcal{A}}$ from the adversary.
- For every $i_1, i_2 \in [n], i_1 \neq i_2$, receive $\{\Pi_{\mathsf{Quad}}[i_1, i_2, 1, 1].\mathsf{msg}_{1,i_4,\to i_5}\}_{i_4 \in \mathcal{A}, i_5 \in \mathcal{H}}$ from the adversary.
- For every $i_1, i_3 \in [n], i_1 \neq i_3, i_2 \in [n+1], j \in [t+1] \backslash \{1\}$, receive $\{\Pi_{\mathsf{Quad}}[i_1, i_2, i_3, j, 1].\mathsf{msg}_{1,i_4,\to i_5}\}_{i_4 \in \mathcal{A}, i_5 \in \mathcal{H}}$ from the adversary.
- For every $i_1, i_2 \in [n], i_1 \neq i_2, j \in [t+1] \setminus \{1\}$, receive $\{\Pi_{\mathsf{Quad}}[i_1, i_2, j, 2].\mathsf{msg}_{1,i_4,\to i_5}\}_{i_4 \in \mathcal{A}, i_5 \in \mathcal{H}}$ from the adversary.
- For every $i_1, i_2 \in [n], i_1 \neq i_2, j \in [t] \setminus \{1\}$, receive $\{\Pi_{\mathsf{DFunc}}[i_1, i_2, j].\mathsf{msg}_{1,i' \to i''}\}_{i \in \mathcal{A}, i'' \in \mathcal{H}}$ from the adversary.

Round 2: $\mathcal{S} \to \mathcal{A}$: The simulator now generates the second round messages on behalf of each honest party $i \in \mathcal{H}$ as follows. Note that at some point in the ideal world execution, when interacting with Sim_Φ, when the simulator Sim_Φ queries the ideal function with the extracted inputs of the adversary, the simulator $\mathcal{S}$ forwards these extracted inputs to the ideal functionality and forwards its response to simulator Sim_Φ. We don't explicitly specify this step anywhere else in the description of the simulator.

- For every $i' \in \mathcal{A}$, the simulator $\mathcal{S}$ sends $\mathbf{Z}^{i',1}$ to Sim_Φ on behalf of party i' and obtains simulated messages $\{\mathbf{Z}^{i,2}[\mathsf{copy1}]\}_{i \in \mathcal{H}}$.
- **Simulating second round messages for** $\Pi_{\mathsf{Quad}}[i_1, i_2, 1, 1]$: For each $i_1, i_2 \in [n], i_1 \neq i_2$, the simulator $\mathcal{S}$ invokes the simulator $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ of Π_{Quad}.
  - For every $i_1 \in \mathcal{A}$
    1. If $i_2 \in \mathcal{H}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality with extracted inputs $\{v_k\}_{k \in \mathbf{L}^o_{i_1,1}}$, $\{R_k\}_{k \in S_{i_1} \bigcup_{i' \in [n] \setminus \{i_1\}} T_{i_1,i'}}$ of the adversary, $\mathcal{S}$ responds with output $\mathbf{K}_{i_2,2}[\mathsf{copy1}]$. Here $\mathbf{K}_{i_2,2}[\mathsf{copy1}]$ is the simulated wire keys for the next garbled circuit of party i_2. It uses the extracted inputs $\{v_k\}_{k \in \mathbf{L}^o_{i_1,1}}, \{R_k\}_{k \in S_{i_1} \bigcup_{i' \in [n] \setminus \{i_1\}} T_{i_1,i'}}$ and $\mathbf{Z}^1$ to compute $\mathbf{Z}^{i_1,2}[\mathsf{copy1}]$ which is consistent with the values of $\mathbf{st}_{i_1}$ committed by the adversary in Round 1.
    2. If $i_2 \in \mathcal{A}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality with extracted inputs $\{v_k\}_{k \in \mathbf{L}^o_{i_1,1}}$, $\{R_k\}_{k \in S_{i_1} \bigcup_{i' \in [n] \setminus \{i_1\}} T_{i_1,i'}}$ and $\mathbf{K}_{i_2,2}[\mathsf{copy1}]$ of the adversary, $\mathcal{S}$ responds with the wire keys in $\mathbf{K}_{i_2,2}[\mathsf{copy1}]$ corresponding to $\{(k, R_k \oplus v_k)\}_{k \in \mathbf{L}^o_{i_1,1}}$.
  - For every $i_1 \in \mathcal{H}$
    1. If $i_2 \in \mathcal{A}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality with extracted inputs $\mathbf{K}_{i_2,2}[\mathsf{copy1}]$ of the adversary, $\mathcal{S}$ responds with the wire keys in $\mathbf{K}_{i_2,2}[\mathsf{copy1}]$ corresponding to $\{(k, z_k)\}_{k \in \mathbf{L}^o_{i_1,1}}$, where $z'_k s$ are updated values in the simulated transcript $\mathbf{Z}^{i_1,2}[\mathsf{copy1}]$.
    2. If $i_2 \in \mathcal{H}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality, $\mathcal{S}$ responds with the wire keys $\mathbf{K}_{i_1,2}[\mathsf{copy1}]$. Here $\mathbf{K}_{i_1,2}[\mathsf{copy1}]$ is the simulated wire keys for the next garbled circuit of party i_2.

  At the end, $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ returns simulated second round messages $\{\Pi_{\mathsf{Quad}}[i_1, i_2, 1, 1].\mathsf{msg}_{2,i_4,\to i_5}\}_{i_4 \in \mathcal{H}, i_5 \in [n]}$ of the honest parties.
- For each $j \in [t+1] \setminus \{1\}$, the simulator $\mathcal{S}$ proceeds as follows:
  - It computes $\mathbf{Z}^j[\mathsf{copy1}]$ as described in fig 2 using the simulated transcript $\{\mathbf{Z}^{i',j}[\mathsf{copy1}]\}_{i' \in \mathcal{H}}$ and the values $\{\mathbf{Z}^{i',j}[\mathsf{copy1}]\}_{i' \in \mathcal{A}}$ computed in the previous step.
  - For every $i' \in \mathcal{A}$, the simulator $\mathcal{S}$ sends $\mathbf{Z}^{i',j}[\mathsf{copy1}]$ to Sim_Φ on behalf of party i' and obtains simulated messages $\{\mathbf{Z}^{i,j}[\mathsf{local}]\}_{i \in \mathcal{H}}$.

– **Simulating second round messages for** $\Pi_{\mathsf{Quad}}[i_1, i_2, i_3, j, 1]$: For each $i_1, i_3 \in [n], i_1 \neq i_3$ and $i_2 \in [n+1]$, the simulator $\mathcal{S}$ invokes the simulator of $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ of Π_{Quad}.

– For every $i_1 \in \mathcal{A}$

1. If $i_3 \in \mathcal{H}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality with extracted inputs $\{R_{k'}\}_{k' \in S_{i_1}}$ of the adversary, $\mathcal{S}$ responds with output $\mathbf{K}_{i_3,j}[\mathsf{local}]$. Here $\mathbf{K}_{i_3,j}[\mathsf{local}]$ is the simulated wire keys for the next garbled circuit of party i_3. It uses the extracted inputs $\{R_{k'}\}_{k' \in S_{i_1}}$, global state $\mathbf{Z}^j[\mathsf{copy1}]$ and $\{R_k\}_{k \in T_{i_2,i_1}}$ (if $i_2 \in \mathcal{A}$, this value is also extracted by $\mathsf{Sim}_{\Pi_{\mathsf{Quad}}}$, else this corresponds to the simulated values generated by Sim_Φ in the first round) to compute $\mathbf{Z}^{i_1,j}[\mathsf{local}]$ which is consistent with the values of $\mathbf{st}_{i_1}$ committed by the adversary in Round 1.
2. If $i_3 \in \mathcal{A}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality with extracted inputs $\{R_{k'}\}_{k' \in S_{i_1}}$ (if $i_2 \in \mathcal{H}$ or $i_2 = n+1$) or $\{R_{k'}\}_{k' \in S_{i_1}}, \{R_k\}_{k \in T_{i_2,i_1}}$ (if $i_2 \in \mathcal{A}$) and $\mathbf{K}_{i_3,j}[\mathsf{local}]$ of the adversary, $\mathcal{S}$ responds with the wire keys 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 defined in fig 2.

– For every $i_1 \in \mathcal{H}$

1. If $i_3 \in \mathcal{A}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality with extracted inputs $\mathbf{K}_{i_2,j}[\mathsf{local}]$ (and additionally $\{R_k\}_{k \in T_{i_2,i_1}}$ if $i_2 \in \mathcal{A}$) of the adversary, $\mathcal{S}$ responds with the wire keys 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 $k' \in \mathbf{L}^{I\leftarrow}_{i_3,j}$ such that k' is the unique location associated with k as defined in fig 2.
2. If $i_3 \in \mathcal{H}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality, $\mathcal{S}$ responds with the wire keys $\mathbf{K}_{i_3,j}[\mathsf{local}]$. Here $\mathbf{K}_{i_3,j}[\mathsf{local}]$ is the simulated wire keys for the next garbled circuit of party i_3.

At the end, $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ returns simulated second round messages $\{\Pi_{\mathsf{Quad}}[i_1, i_2, i_3, j, 1].\mathsf{msg}_{2,i_4,\rightarrow i_5}\}_{i_4 \in \mathcal{H}, i_5 \in [n]}$ for the honest parties.

– It computes $\mathbf{Z}^j[\mathsf{local}]$ as described in fig 3 using the simulated transcript $\{\mathbf{Z}^{i',j}[\mathsf{local}]\}_{i' \in \mathcal{H}}$ and the values $\{\mathbf{Z}^{i',j}[\mathsf{local}]\}_{i' \in \mathcal{A}}$ computed in the previous step.

– For every $i' \in \mathcal{A}$, the simulator $\mathcal{S}$ sends $\mathbf{Z}^{i',j}[\mathsf{local}]$ to Sim_Φ on behalf of party i' and obtains simulated messages $\{\mathbf{Z}^{i,j}[\mathsf{copy2}]\}_{i \in \mathcal{H}}$.

– **Simulating second round messages for** $\Pi_{\mathsf{DFunc}}[i_1, i_2, j]$: The function f associated with the functionality $DF_{i_1,i_2,j}$ for $\Pi_{\mathsf{DFunc}}[i_1, i_2, j]$ is

$$f = \Phi_{i_2,j}(\mathbf{Z}^j[\mathsf{local}], .)$$

For every $i_1, i_2 \in [n], i_1 \neq i_2$, the simulator invokes the simulator $\mathsf{Sim}^2_{\Pi_{\mathsf{DFunc}}}$ of Π_{DFunc} with the function f.

* For every $i_1 \in \mathcal{H}$
  1. If $i_2 \in \mathcal{A}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{DFunc}}}$ queries the Ideal functionality with extracted implicit input $\{(k, z_k)\}_{k \in S_{i_2}}$ of the adversary, where $\{(k, z_k)\}_{k \in S_{i_2}} = f(\{R_k\}_{k \in S_{i_2}})$, $\mathcal{S}$ responds with output $\mathbf{K}_{i_1,j}[\mathsf{copy2}]$. Here $\mathbf{K}_{i_1,j}[\mathsf{copy2}]$ is the simulated wire keys for the next garbled circuit of party i_1. It uses the extracted inputs $\{(k, z_k)\}_{k \in S_{i_2}}$ and $\mathbf{Z}^j[\mathsf{local}]$ to compute $\mathbf{Z}^{i_2,j}[\mathsf{copy2}]$ which is consistent with the values of $\mathbf{st}_{i_2}$ committed by the adversary in Round 1.
  2. If $i_2 \in \mathcal{H}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{DFunc}}}$ queries the Ideal functionality, $\mathcal{S}$ responds with the wire keys $\mathbf{K}_{i_1,j}[\mathsf{copy2}]$. Here $\mathbf{K}_{i_1,j}[\mathsf{copy2}]$ is the simulated wire keys for the next garbled circuit of party i_1.
* For every $i_1 \in \mathcal{A}$

1. If $i_2 \in \mathcal{H}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{DFunc}}}$ queries the Ideal functionality with extracted inputs $\mathbf{K}_{i_1,j}[\mathsf{copy2}]$ of the adversary, $\mathcal{S}$ responds with the wire keys in $\mathbf{K}_{i_1,j}[\mathsf{copy2}]$ corresponding to $\{(k,z_k)\}_{k\in S_{i_2}}$, where $z'_k s$ are updated values in the simulated transcript $\mathbf{Z}^{i_2,j}[\mathsf{copy2}]$.
2. If $i_2 \in \mathcal{A}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{DFunc}}}$ queries the Ideal functionality with the extracted values $\mathbf{K}_{i_1,j}[\mathsf{copy2}]$ and $\{(k,z_k)\}_{k\in S_{i_2}}$, where $\{(k,z_k)\}_{k\in S_{i_2}} = f(\{R_k\}_{k\in S_{i_2}})$ $\mathcal{S}$ responds with the wire keys in $\mathbf{K}_{i_1,j}[\mathsf{copy2}]$ corresponding to $\{(k,z_k)\}_{k\in S_{i_2}}$.

At the end, $\mathsf{Sim}^2_{\Pi_{\mathsf{DFunc}}}$ returns simulated second round messages $\{\Pi_{\mathsf{DFunc}}[i_1,i_2,j].\mathsf{msg}_{2,i'\to i''}\}_{i\in\mathcal{H},i''\in\mathcal{A}}$ of the honest parties.

- It computes $\mathbf{Z}^j[\mathsf{copy2}]$ as described in fig 4 using the simulated transcript $\{\mathbf{Z}^{i',j}[\mathsf{copy2}]\}_{i'\in\mathcal{H}}$ and the values $\{\mathbf{Z}^{i',j}[\mathsf{copy2}]\}_{i'\in\mathcal{A}}$ computed in the previous step.
- If $j \neq t+1$, for every $i' \in \mathcal{A}$, the simulator $\mathcal{S}$ sends $\mathbf{Z}^{i',j}[\mathsf{copy2}]$ to Sim_Φ on behalf of party i' and obtains simulated messages $\{\mathbf{Z}^{i,j+1}[\mathsf{copy1}]\}_{i\in\mathcal{H}}$.
- **Simulating second round messages for** $\Pi_{\mathsf{Quad}}[i_1,i_2,j,2]$: If $j \neq t+1$, for each $i_1,i_2 \in [n], i_1 \neq i_2$, the simulator $\mathcal{S}$ invokes the simulator of $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ of Π_{Quad}.
- For every $i_1 \in \mathcal{A}$
  1. If $i_2 \in \mathcal{H}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality with extracted inputs $\{R_{k'}\}_{k'\in S_{i_1}}$, $\{\{R_k\}_{k\in T_{i_1,i'}}\}_{i'\in[n]\setminus\{i_1\}}$ of the adversary, $\mathcal{S}$ responds with output $\mathbf{K}_{i_2,j+1}[\mathsf{copy1}]$. Here $\mathbf{K}_{i_2,j+1}[\mathsf{copy1}]$ is the simulated wire keys for the next garbled circuit of party i_2. It uses the extracted inputs $\{R_{k'}\}_{k'\in S_{i_1}}, \{\{R_k\}_{k\in T_{i_1,i'}}\}_{i'\in[n]\setminus\{i_1\}}$ and $\mathbf{Z}^j[\mathsf{local}]$ to compute $\mathbf{Z}^{i_1,j+1}[\mathsf{copy1}]$ which is consistent with the values of $\mathbf{st}_{i_1}$ committed by the adversary in Round 1.
  2. If $i_2 \in \mathcal{A}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality with extracted inputs $\{R_{k'}\}_{k'\in S_{i_1}}$, $\{\{R_k\}_{k\in T_{i_1,i'}}\}_{i'\in[n]\setminus\{i_1\}}$ and $\mathbf{K}_{i_2,j+1}[\mathsf{copy1}]$ of the adversary, $\mathcal{S}$ responds with the wire keys in $\mathbf{K}_{i_2,j+1}[\mathsf{copy1}]$ corresponding to $\{(k', R'_k \oplus R_k \oplus z_k)\}$, for every location $k' \in \mathbf{L}^{O\to}_{i_1,j}$ and $k \in \mathbf{L}^{O\leftarrow}_{i_1,j}$ such that k' is the unique location associated with k as defined in fig 4.
- For every $i_1 \in \mathcal{H}$
  1. If $i_2 \in \mathcal{A}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality with extracted inputs $\mathbf{K}_{i_2,j+1}[\mathsf{copy1}]$ of the adversary, $\mathcal{S}$ responds with the wire keys in $\mathbf{K}_{i_2,j+1}[\mathsf{copy1}]$ corresponding to $\{(k', z_k)\}$, for every location $k \in \mathbf{L}^{O\leftarrow}_{i_1,j}$ where $z'_k s$ are the updated values in the simulated transcript $\mathbf{Z}^{i_1,j+1}[\mathsf{copy1}]$
  2. If $i_2 \in \mathcal{H}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality, $\mathcal{S}$ responds with the wire keys $\mathbf{K}_{i_2,j+1}[\mathsf{copy1}]$. Here $\mathbf{K}_{i_2,j+1}[\mathsf{copy1}]$ is the simulated wire keys for the next garbled circuit of party i_2.

Finally, $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ returns simulated second round messages $\{\Pi_{\mathsf{Quad}}[i_1,i_2,j,2].\mathsf{msg}_{2,i_4,\to i_5}\}_{i_4\in\mathcal{H},i_5\in[n]}$ for the honest parties.

- For each $i \in \mathcal{H}$, the simulator invokes the $\mathsf{Sim}^2_{\mathsf{GC}}$ to simulate second set of garbled circuit messages.

$$\mathsf{GC}_{i,1} \leftarrow \mathsf{Sim}^2_{\mathsf{GC}}(\mathsf{gk}_{i,1}, \mathbf{K}_{i,1}, (\mathbf{K}_{i,2}[\mathsf{copy1}][\mathbf{Z}^{i,2}[\mathsf{copy1}]], \{\Pi_{\mathsf{Quad}}[i_1,i_2,1,1].\mathsf{msg}_{2,i\to i'}\}_{i_1,i_2,i'\in[n]}))$$

where $\mathbf{Z}^{i,2}[\mathsf{copy1}]$ is the simulated intermediate state.

- For each $j \in [t+1]\setminus\{1\}$ and $i \in \mathcal{H}$, the simulator invokes the $\mathsf{Sim}_{\mathsf{GC}}$ to simulate second set of garbled circuit messages.

$$\mathsf{GC}_{i,j}[\mathsf{copy1}] \leftarrow$$
$$\mathsf{Sim}^2_{\mathsf{GC}}(\mathsf{gk}_{i,j}[\mathsf{copy1}], \mathbf{K}_{i,j}[\mathsf{copy1}], (\mathbf{K}_{i,j}[\mathsf{local}][\mathbf{Z}^{i,j}[\mathsf{local}]], \{\Pi_{\mathsf{Quad}}[i_1,i_2,i_3,j,1].\mathsf{msg}_{2,i\to i'}\}_{i_1,i_2,i_3,i'\in[n]}))$$

$$\mathsf{GC}_{i,j}[\mathsf{local}] \leftarrow \mathsf{Sim}^2_{\mathsf{GC}}(\mathsf{gk}_{i,j}[\mathsf{local}], \mathbf{K}_{i,j}[\mathsf{local}], (\mathbf{K}_{i,j}[\mathsf{copy2}][\mathbf{Z}^{i,j}[\mathsf{copy2}]], \{\Pi_{\mathsf{DFunc}}[i_1,i_2,j].\mathsf{msg}_{2,i\to i'}\}_{i_1,i_2,i'\in[n]}))$$

If $j = t+1$

$$\mathsf{GC}_{i,j}[\mathsf{copy2}] \leftarrow \mathsf{Sim}^2_{\mathsf{GC}}(\mathsf{gk}_{i,j}[\mathsf{copy2}], \mathbf{K}_{i,j}[\mathsf{copy2}], \mathbf{Z}^{t+1}[\mathsf{copy2}])$$

Else if $j \neq t+1$

$$\mathsf{GC}_{i,j}[\mathsf{copy2}] \leftarrow$$
$$\mathsf{Sim}^2_{\mathsf{GC}}(\mathsf{gk}_{i,j}[\mathsf{copy2}], \mathbf{K}_{i,j}[\mathsf{copy2}], (\mathbf{K}_{i,j+1}[\mathsf{copy1}][\mathbf{Z}^{i,j+1}[\mathsf{copy1}]], \{\Pi_{\mathsf{Quad}}[i_1, i_2, j, 2].\mathsf{msg}_{2,i\to i'}\}_{i_1,i_2,i_3,i'\in[n]}))$$

- The simulator sends the following to the adversary:

$$\{\mathsf{GC}_{i,1}, \mathbf{K}_{i,1}, \{\mathsf{GC}_{i,j}[\mathsf{copy1}], \mathsf{GC}_{i,j}[\mathsf{local}], \mathsf{GC}_{i,j}[\mathsf{copy2}]\}_{j\in[n]}\}_{i\in\mathcal{H}}$$

We prove security via a sequence of hybrids, where $\mathbf{H}_1$ is the real execution and $\mathbf{H}_4$ is the simulated execution.

$\mathbf{H}_1$: Execution of the protocol in the real world.

$\mathbf{H}_{2.1.1}$ Identical to $\mathbf{H}_1$, except that we simulate the garbled circuits corresponding to the 1^{st} round of the computation phase that are sent by the honest parties. This additionally involves simulating/pre-computing the messages that the garbled circuit outputs.

We now simulate the garbled circuits corresponding to the first round of the underlying protocol using the transcript computed in the previous hybrid.

For each $i \in \mathcal{H}$, in Round 1,

$$\{(\mathsf{gk}_{i,1}, \mathbf{K}_{i,1})\}_{i\in\mathcal{H}} \leftarrow \{\mathsf{Sim}^1_{\mathsf{GC}}(1^{\mathbf{k}}, \varphi(C_{i,1}))\}_{i\in\mathcal{H}}$$

where $\varphi(C_{i,1})$ is the topology of the circuit in Figure 1. and in round 2,

$$\mathsf{GC}_{i,1}, \leftarrow \mathsf{Sim}^2_{\mathsf{GC}}(\mathsf{gk}_{i,1}, \mathbf{K}_{i,1}, (\mathbf{K}_{i,2}[\mathsf{copy1}][\mathbf{Z}^{i,2}[\mathsf{copy1}]], \{\Pi_{\mathsf{Quad}}[i_1, i_2, 1, 1].\mathsf{msg}_{2,i\to i'}\}_{i_1,i_2,i'\in[n]}))$$

where $\mathbf{K}_{i,2}[\mathsf{copy1}][\mathbf{Z}^{i,2}[\mathsf{copy1}]]$ are the honestly generated labels for $\mathsf{GC}_{i,2}[\mathsf{copy1}]$ corresponding to the intermediate global state $\mathbf{Z}^{i,2}[\mathsf{copy1}]$ computed using the first round messages $\{\mathbf{Z}^{i',1}\}_{i'\in[n]}$ and the inputs of the honest parties.

Claim 12. *From statistical security of the garbling scheme, hybrids $\mathbf{H}_1$ and $\mathbf{H}_{2.1.1}$ are statistically indistinguishable.*

Proof. The statistical indistinguishability of hybrids $\mathbf{H}_2$ and $\mathbf{H}_{2.1.1}$ follows from statistical security of the garbled circuits. The reduction follows from $|\mathcal{H}|(> \frac{n+1}{2})$ invocations of security of the garbled circuit. □

$\mathbf{H}_{2.1.2}$ This hybrid is similar to $\mathbf{H}_{2.1.1}$ except that we now simulate the messages of honest parties for the Π_{Quad} protocols corresponding to the first round of the underlying conforming protocol.

We start by simulating the first round messages that are sent to the adversary by the honest parties.

For every $i_1, i_2 \in [n], i_1 \neq i_2$, the simulator invokes the simulator $\mathsf{Sim}_{\Pi_{\mathsf{Quad}}}$ of Π_{Quad} to obtain simulated first round messages of the honest parties.

$$\{\Pi_{\mathsf{Quad}}[i_1, i_2, 1, 1].\mathsf{msg}_{1,i_4,\to i_5}\}_{i_4\in\mathcal{H}, i_5\in\mathcal{A}} \leftarrow \mathsf{Sim}^1_{\Pi_{\mathsf{Quad}}}(1^{\mathbf{k}})$$

Then in the second round, we invoke the simulator $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ of Π_{Quad}.

- For every $i_1 \in \mathcal{A}$

1. If $i_2 \in \mathcal{H}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality with extracted inputs $\{v_k\}_{k\in\mathbf{L}^{\mathcal{O}}_{i_1,1}}$, $\{R_k\}_{k\in S_{i_1}\bigcup_{i'\in[n]\setminus\{i_1\}}T_{i_1,i'}}$ of the adversary, $\mathcal{S}$ responds with the wire keys in $\mathbf{K}_{i_2,2}[\mathsf{copy1}]$ corresponding to $\{(k, R_k \oplus v_k)\}_{k\in\mathbf{L}^{\mathcal{O}}_{i_1,1}}$. Here $\mathbf{K}_{i_2,2}[\mathsf{copy1}]$ are the wire keys for the next garbled circuit of party i_2. It uses the extracted inputs $\{v_k\}_{k\in\mathbf{L}^{\mathcal{O}}_{i_1,1}}, \{R_k\}_{k\in S_{i_1}\bigcup_{i'\in[n]\setminus\{i_1\}}T_{i_1,i'}}$ and $\mathbf{Z}^1$ to compute $\mathbf{Z}^{i_1,2}[\mathsf{copy1}]$ which is consistent with the values of $\mathbf{st}_{i_1}$ committed by the adversary in Round 1.
2. If $i_2 \in \mathcal{A}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality with extracted inputs $\{v_k\}_{k\in\mathbf{L}^{\mathcal{O}}_{i_1,1}}$, $\{R_k\}_{k\in S_{i_1}\bigcup_{i'\in[n]\setminus\{i_1\}}T_{i_1,i'}}$ and $\mathbf{K}_{i_2,2}[\mathsf{copy1}]$ of the adversary, $\mathcal{S}$ responds with the wire keys in $\mathbf{K}_{i_2,2}[\mathsf{copy1}]$ corresponding to $\{(k, R_k \oplus v_k)\}_{k\in\mathbf{L}^{\mathcal{O}}_{i_1,1}}$.

- For every $i_1 \in \mathcal{H}$
  1. If $i_2 \in \mathcal{A}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality with extracted inputs $\mathbf{K}_{i_2,2}[\mathsf{copy1}]$ of the adversary, $\mathcal{S}$ responds with the wire keys in $\mathbf{K}_{i_1,2}[\mathsf{copy1}]$ corresponding to $\{(k, z_k)\}_{k\in\mathbf{L}^{\mathcal{O}}_{i_1,1}}$, where $z'_k s$ are updated values in the honestly computed transcript $\mathbf{Z}^{i_1,2}[\mathsf{copy1}]$.
  2. If $i_2 \in \mathcal{H}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality, $\mathcal{S}$ responds with the wire keys in $\mathbf{K}_{i_2,2}[\mathsf{copy1}]$ corresponding to $\{(k, z_k)\}_{k\in\mathbf{L}^{\mathcal{O}}_{i_1,1}}$, where $z'_k s$ are updated values in the honestly computed transcript $\mathbf{Z}^{i_1,2}[\mathsf{copy1}]$. Here $\mathbf{K}_{i_2,2}[\mathsf{copy1}]$ are the wire keys for the next garbled circuit of party i_2.

At the end, $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ returns simulated second round messages $\{\Pi_{\mathsf{Quad}}[i_1, i_2, 1, 1].\mathsf{msg}_{2,i_4,\to i_5}\}_{i_4\in\mathcal{H},i_5\in[n]}$ for the honest parties. The simulator $\mathcal{S}$ computes the rest of the messages as in Hybrid $\mathbf{H}_{2.1.1}$ and sends them to the adversary on behalf of the honest parties.

After receiving second round messages from the adversary, the simulator $\mathcal{S}$ evaluates the garbled circuits $\{\mathsf{GC}_{i',1}\}_{i'\in\mathcal{A}}$ of the adversary to obtain second round messages for $\Pi_{\mathsf{Quad}}[i_1, i_2, 1, 1]$. It sends these to the simulator $\mathsf{Sim}_{\Pi_{\mathsf{Quad}}}$ to obtain the output $y[i_1, i_2, 1, 1]$ of the honest parties. For $i_2 \in \mathcal{H}$ and $y[i_1, i_2, 1, 1]$ corresponds to the wire keys in $\mathbf{K}_{i_1,2}[\mathsf{copy1}]$ for any value other than $\{(k, z_k)\}_{k\in\mathbf{L}^{\mathcal{O}}_{i_1,1}}$, then $\mathcal{S}$ sends $\perp$ to $\mathsf{Sim}_{\Pi_{\mathsf{Quad}}}$ as the output for the honest parties in $\Pi_{\mathsf{Quad}}[i_1, i_2, 1, 1]$. In this case $\mathcal{S}$ also sets the final output of the honest parties to $\perp$. In all other cases it sends $y[i_1, i_2, 1, 1]$ to $\mathsf{Sim}_{\Pi_{\mathsf{Quad}}}$ as the output for the honest parties in $\Pi_{\mathsf{Quad}}[i_1, i_2, 1, 1]$, it evaluates the final output as in the previous hybrid.

Claim 13. *From statistical privacy with knowledge of outputs security of Π_{Quad}, hybrids $\mathbf{H}_{2.1.2}$ and $\mathbf{H}_{2.1.1}$ are statistically indistinguishable.*

Proof. The indistinguishability of hybrids $\mathbf{H}_{2.1.2}$ and $\mathbf{H}_{2.1.1}$ follows from a sequence of $n(n-1)$ further sub-hybrids, where in each sub-hybrid, we change one Π_{Quad} execution between parties from an honest execution to a simulated execution. □

Similarly for $j \in \{2, \ldots, t+1\}$, we have the following sequence of hybrids:

$\mathbf{H}_{3.j.1.1}$ Similar to hybrid $\mathbf{H}_{3.j-1.3.2}$(or $\mathbf{H}_{2.1.2}$ if $j = 2$) except that now we simulate the garbled circuits corresponding to the $\mathsf{copy1}$ function of the $j-$th round of the computation phase that are sent by the honest parties. We compute $\mathbf{Z}^j[\mathsf{copy1}]$ using $\{\mathbf{Z}^{i',j}[\mathsf{copy1}]\}_{i'\in\mathcal{H}}$ and the value $\{\mathbf{Z}^{i',j}[\mathsf{copy1}]\}_{i'\in\mathcal{A}}$ computed as in the previous hybrid. Statistical indistinguishability between these two hybrids follows similar to the statistical indistinguishability between $\mathbf{H}_1$ and $\mathbf{H}_{2.1.1}$.

$\mathbf{H}_{3.j.1.2}$ Similar to hybrid $\mathbf{H}_{3.j.1.1}$ except that we now simulate the messages of honest parties for the Π_{Quad} protocols corresponding to the first copy function of the $j-$th round of the underlying protocol as follows:

For each $i_1, i_3 \in [n], i_1 \neq i_3$ and $i_2 \in [n+1]$, the simulator $\mathcal{S}$ uses $\mathsf{Sim}^1_{\Pi_{\mathsf{Quad}}}$ to simulate the first round messages for $\Pi_{\mathsf{Quad}}[i_1, 1_2, i_3, j, 1]$. It then uses $\{\mathbf{Z}^{i',j}[\mathsf{copy1}]\}_{i' \in \mathcal{A}}$ to compute messages $\{\mathbf{Z}^{i,j}[\mathsf{local}]\}_{i \in \mathcal{H}}$ for the honest parties. The simulator $\mathcal{S}$ then invokes the simulator $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ of Π_{Quad}.

- For every $i_1 \in \mathcal{A}$
  1. If $i_3 \in \mathcal{H}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality with extracted inputs $\{R_{k'}\}_{k' \in S_{i_1}}$ of the adversary, $\mathcal{S}$ responds with the wire keys 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 defined in fig 2. Here $\mathbf{K}_{i_3,j}[\mathsf{local}]$ are the wire keys for the next garbled circuit of party i_3. It uses the extracted inputs $\{R_{k'}\}_{k' \in S_{i_1}}$, global state $\mathbf{Z}^j[\mathsf{copy1}]$ and values $\{R_k\}_{k \in T_{i_2,i_1}}$ (if $i_2 \in \mathcal{A}$, this value is also extracted by $\mathsf{Sim}_{\Pi_{\mathsf{Quad}}}$, else this corresponds to the simulated values generated by Sim_Φ in the first round) to compute $\mathbf{Z}^{i_1,j}[\mathsf{local}]$ which is consistent with the values of $\mathbf{st}_{i_1}$ committed by the adversary in Round 1.
  2. If $i_3 \in \mathcal{A}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality with extracted inputs $\{R_{k'}\}_{k' \in S_{i_1}}$ (if $i_2 \in \mathcal{H}$ or $i_2 = n+1$) or $\{R_{k'}\}_{k' \in S_{i_1}}, \{R_k\}_{k \in T_{i_2,i_1}}$ (if $i_2 \in \mathcal{A}$) and $\mathbf{K}_{i_3,j}[\mathsf{local}]$ of the adversary, $\mathcal{S}$ responds with the wire keys 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 defined in fig 2.
- For every $i_1 \in \mathcal{H}$
  1. If $i_3 \in \mathcal{A}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality with extracted inputs $\mathbf{K}_{i_2,j}[\mathsf{local}]$ (and additionally $\{R_k\}_{k \in T_{i_2,i_1}}$ if $i_2 \in \mathcal{A}$) of the adversary, $\mathcal{S}$ responds with the wire keys 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 $k' \in \mathbf{L}^{I\leftarrow}_{i_3,j}$ such that k' is the unique location associated with k as defined in fig 2.
  2. If $i_3 \in \mathcal{H}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality, $\mathcal{S}$ responds with the wire keys $\mathbf{K}_{i_3,j}[\mathsf{local}]$ 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 defined in fig 2. Here $\mathbf{K}_{i_3,j}[\mathsf{local}]$ are the wire keys for the next garbled circuit of party i_3.

At the end, $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ returns simulated second round messages $\{\Pi_{\mathsf{Quad}}[i_1, i_2, i_3, j, 1].\mathsf{msg}_{2,i_4,\rightarrow i_5}\}_{i_4 \in \mathcal{H}, i_5 \in [n]}$ for the honest parties.

Statistical indistinguishability between these two hybrids follows similar to the statistical indistinguishability between $\mathbf{H}_{2.1.2}$ and $\mathbf{H}_{2.1.1}$.

$\mathbf{H}_{3.j.2.1}$ Similar to hybrid $\mathbf{H}_{3.j.1.2}$ except that now we simulate the garbled circuits corresponding to the local function of the j-th round of the computation phase that are sent by the honest honest parties. We compute $\mathbf{Z}^j[\mathsf{local}]$ using $\{\mathbf{Z}^{i',j}[\mathsf{local}]\}_{i' \in \mathcal{H}}$ and the value $\{\mathbf{Z}^{i',j}[\mathsf{local}]\}_{i' \in \mathcal{A}}$ computed as in the previous hybrid.

Statistical indistinguishability between these two hybrids follows similar to the statistical indistinguishability between $\mathbf{H}_2$ and $\mathbf{H}_{2.1.1}$.

$\mathbf{H}_{3.j.2.2}$ Similar to hybrid $\mathbf{H}_{3.j.2.1}$ except that we now simulate the messages of honest parties for the Π_{DFunc} protocols corresponding to the local function of the $j-$th round of the underlying protocol as follows:

For every $i_1, i_2 \in [n], i_1 \neq i_2$ $j \in [t+1]\backslash$, the simulator $\mathcal{S}$ uses $\mathsf{Sim}^1_{\Pi_{\mathsf{DFunc}}}$ to simulate the first round messages for $\Pi_{\mathsf{DFunc}}[i_1, i_2, j]$. It then uses $\{\mathbf{Z}^{i',j}[\mathsf{local}]\}_{i' \in \mathcal{A}}$ to compute messages $\{\mathbf{Z}^{i,j}[\mathsf{copy2}]\}_{i \in \mathcal{H}}$ for the honest parties. The function f associated with the functionality $DF_{i_1,i_2,j}$ for $\Pi_{\mathsf{DFunc}}[i_1, i_2, j]$ is

$$f = \Phi_{i_2,j}(\mathbf{Z}^j[\mathsf{local}], .)$$

For every $i_1, i_2 \in [n], i_1 \neq i_2$ $j \in [t+1]\backslash$, the simulator invokes the simulator $\mathsf{Sim}^2_{\Pi_{\mathsf{DFunc}}}$ of Π_{DFunc} with the function f.

- For every $i_1 \in \mathcal{H}$
  1. If $i_2 \in \mathcal{A}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{DFunc}}}$ queries the Ideal functionality with extracted implicit inputs $\{(k, z_k)\}_{k \in S_{i_2}}$ of the adversary, where $\{(k, z_k)\}_{k \in S_{i_2}} = f(\{R_k\}_{k \in S_{i_2}})$. $\mathcal{S}$ responds with the wire keys in $\mathbf{K}_{i_1,j}[\mathsf{copy2}]$ corresponding to $\{(k, z_k)\}_{k \in S_{i_2}}$. Here $\mathbf{K}_{i_1,j}[\mathsf{copy2}]$ are the wire keys for the next garbled circuit of party i_1. It uses the extracted inputs $\{(k, z_k)\}_{k \in S_{i_2}}$ and $\mathbf{Z}^j[\mathsf{local}]$ to compute $\mathbf{Z}^{i_2,j}[\mathsf{copy2}]$ which is consistent with the values of $\mathbf{st}_{i_2}$ committed by the adversary in Round 1.
  2. If $i_2 \in \mathcal{H}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{DFunc}}}$ queries the Ideal functionality, $\mathcal{S}$ responds with the wire keys in $\mathbf{K}_{i_1,j}[\mathsf{copy2}]$ corresponding to $\{(k, z_k)\}_{k \in S_{i_2}}$, where $z'_k s$ are updated values in the simulated transcript $\mathbf{Z}^{i_2,j}[\mathsf{copy2}]$. Here $\mathbf{K}_{i_1,j}[\mathsf{copy2}]$ are the wire keys for the next garbled circuit of party i_1.
- For every $i_1 \in \mathcal{A}$
  1. If $i_2 \in \mathcal{H}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{DFunc}}}$ queries the Ideal functionality with extracted inputs $\mathbf{K}_{i_1,j}[\mathsf{copy2}]$ of the adversary, $\mathcal{S}$ responds with the wire keys in $\mathbf{K}_{i_1,j}[\mathsf{copy2}]$ corresponding to $\{(k, z_k)\}_{k \in S_{i_2}}$, where $z'_k s$ are updated values in the simulated transcript $\mathbf{Z}^{i_2,j}[\mathsf{copy2}]$.
  2. If $i_2 \in \mathcal{A}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{DFunc}}}$ queries the Ideal functionality with the extracted values $\mathbf{K}_{i_1,j}[\mathsf{copy2}]$ and $\{(k, z_k)\}_{k \in S_{i_2}}$, where $\{(k, z_k)\}_{k \in S_{i_2}} = f(\{R_k\}_{k \in S_{i_2}})$. $\mathcal{S}$ responds with the wire keys in $\mathbf{K}_{i_1,j}[\mathsf{copy2}]$ corresponding to $\{(k, z_k)\}_{k \in S_{i_2}}$.

At the end, $\mathsf{Sim}^2_{\Pi_{\mathsf{DFunc}}}$ returns simulated second round messages $\{\Pi_{\mathsf{DFunc}}[i_1, i_2, j].\mathsf{msg}_{2,i' \to i''}\}_{i \in \mathcal{H}, i'' \in \mathcal{A}}$ of the honest parties. Statistical indistinguishability between these two hybrids follows similar to the statistical indistinguishability between $\mathbf{H}_{2.1.2}$ and $\mathbf{H}_{2.1.1}$.

$\mathbf{H}_{3.j.3.1}$ Similar to hybrid $\mathbf{H}_{3.j.2.2}$ except that now we simulate the garbled circuits corresponding to the $\mathsf{copy2}$ function of the $j-$th round of the computation phase that are sent by the honest honest parties. We also compute $\mathbf{Z}^j[\mathsf{copy2}]$ using $\{\mathbf{Z}^{i',j}[\mathsf{copy2}]\}_{i' \in \mathcal{H}}$ and the value $\{\mathbf{Z}^{i',j}[\mathsf{copy2}]\}_{i' \in \mathcal{A}}$ computed as in the previous hybrid. Statistical indistinguishability between these two hybrids follows similar to the Statistical indistinguishability between $\mathbf{H}_1$ and $\mathbf{H}_{2.1.1}$.

$\mathbf{H}_{3.j.3.2}$ If $j = t+1$, this hybrid is identical to hybrid $\mathbf{H}_{3.j.3.1}$. Else, it is similar to hybrid $\mathbf{H}_{3.j.3.1}$ except that we now simulate the messages of honest parties for the Π_{Quad} protocols corresponding to the second copy function of the $j-$th round of the underlying protocol as follows:

For each $i_1, i_3 \in [n], i_1 \neq i_3$ and $i_2 \in [n+1]$, the simulator $\mathcal{S}$ uses $\mathsf{Sim}^1_{\Pi_{\mathsf{Quad}}}$ to simulate first round messages for $\Pi_{\mathsf{Quad}}[i_1, i_2, j, 2]$. It then uses$\{\mathbf{Z}^{i',j}[\mathsf{copy2}]\}_{i' \in \mathcal{A}}$ to compute messages $\{\mathbf{Z}^{i,j+1}[\mathsf{copy1}]\}_{i \in \mathcal{H}}$ for the honest parties.

For each $i_1, i_2 \in [n], i_1 \neq i_2$ the simulator $\mathcal{S}$ invokes the simulator of $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ of Π_{Quad}.

- For every $i_1 \in \mathcal{A}$
  1. If $i_2 \in \mathcal{H}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality with extracted inputs $\{R_{k'}\}_{k' \in S_{i_1}}$, $\{\{R_k\}_{k \in T_{i_1,i'}}\}_{i' \in [n] \setminus \{i_1\}}$ of the adversary, $\mathcal{S}$ responds with the wire keys in $\mathbf{K}_{i_2,j+1}[\mathsf{local}]$ corresponding to $\{(k', R'_k \oplus R_k \oplus z_k)\}$, for every location $k' \in \mathbf{L}^{O \to}_{i_1,j}$ and $k \in \mathbf{L}^{O \leftarrow}_{i_1,j}$ such that k' is the unique location associated with k as defined in fig 4. Here $\mathbf{K}_{i_2,j+1}[\mathsf{local}]$ are the wire keys for the next garbled circuit of party i_2. It uses the extracted inputs $\{R_{k'}\}_{k' \in S_{i_1}}, \{\{R_k\}_{k \in T_{i_1,i'}}\}_{i' \in [n] \setminus \{i_1\}}$ and $\mathbf{Z}^j[\mathsf{local}]$ to compute $\mathbf{Z}^{i_1,j+1}[\mathsf{copy1}]$ which is consistent with the values of $\mathbf{st}_{i_1}$ committed by the adversary in Round 1.
  2. If $i_2 \in \mathcal{A}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality with extracted inputs $\{R_{k'}\}_{k' \in S_{i_1}}$, $\{\{R_k\}_{k \in T_{i_1,i'}}\}_{i' \in [n] \setminus \{i_1\}}$ and $\mathbf{K}_{i_2,j+1}[\mathsf{copy1}]$ of the adversary, $\mathcal{S}$ responds with the wire keys in $\mathbf{K}_{i_2,j+1}[\mathsf{copy1}]$ corresponding to $\{(k', R'_k \oplus R_k \oplus z_k)\}$, for every location $k' \in \mathbf{L}^{O \to}_{i_1,j}$ and $k \in \mathbf{L}^{O \leftarrow}_{i_1,j}$ such that k' is the unique location associated with k as defined in fig 4.

- For every $i_1 \in \mathcal{H}$
  1. If $i_2 \in \mathcal{A}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality with extracted inputs $\mathbf{K}_{i_2,j+1}[\mathsf{copy1}]$ of the adversary, $\mathcal{S}$ responds with the wire keys in $\mathbf{K}_{i_2,j+1}[\mathsf{copy1}]$ corresponding to $\{(k', z_k)\}$, for every location $k \in \mathbf{L}^{O\leftarrow}_{i_1,j}$ where $z'_k s$ are the updated values in the simulated transcript $\mathbf{Z}^{i_1,j+1}[\mathsf{copy1}]$
  2. If $i_2 \in \mathcal{H}$, when $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ queries the Ideal functionality, $\mathcal{S}$ responds with the wire keys in $\mathbf{K}_{i_2,j+1}[\mathsf{copy1}]$ corresponding to $\{(k', z_k)\}$, for every location $k \in \mathbf{L}^{O\leftarrow}_{i_1,j}$ where $z'_k s$ are the updated values in the simulated transcript $\mathbf{Z}^{i_1,j+1}[\mathsf{copy1}]$. Here $\mathbf{K}_{i_2,j+1}[\mathsf{copy1}]$ is the simulated wire keys for the next garbled circuit of party i_2.

At the end, $\mathsf{Sim}^2_{\Pi_{\mathsf{Quad}}}$ returns simulated second round messages $\{\Pi_{\mathsf{Quad}}[i_1, i_2, j, 2].\mathsf{msg}_{2,i_4,\to i_5}\}_{i_4\in\mathcal{H},i_5\in[n]}$ for the honest parties. Statistical indistinguishability between these two hybrids follows similar to the statistical indistinguishability between $\mathbf{H}_{2.1.2}$ and $\mathbf{H}_{2.1.1}$.

$\mathbf{H}_4$ Similar to hybrid $\mathbf{H}_{3.t+1.3.2}$ except that instead of using the inputs of the honest parties to compute their transcripts in the conforming protocol, we rely on the simulator of the underlying protocol, Sim_Φ, to simulate the messages of the honest parties.

Claim 14. *From the statistical malicious security of the underlying conforming protocol Φ, $\mathbf{H}_4$ and $\mathbf{H}_{3.t+1.3.2}$ are statistically indistinguishable.*

Proof. The statistical indistinguishability of hybrids $\mathbf{H}_{3.t+1.3.2}$ and Hyb_4 follows from the malicious security of the underlying conforming protocol Φ. □

Hyb_4 is identical to the ideal world simulator, and hence we're done.

5.2 From Privacy with Knowledge of Outputs to Security with Abort

We now show that existence of a maliciously secure two-round IT-MPC protocol for functions in NC^1 that achieves privacy with knowledge of outputs, implies existence of a maliciously secure two-round IT-MPC protocol for functions in NC^1 that achieves security with abort. This reduction preserves the corruption threshold. We observe that this transformation can be generalized and works for any arbitrary round protocol.

Lemma 5.3. *There exists a general complexity preserving information-theoretic compiler that transforms any two-round MPC that achieves strong privacy with knowledge of outputs into a two-round protocol that achieves security with abort against malicious adversaries with the same corruption threshold.*

Building Blocks. We use the following ingredients in our construction.

- A two round MPC protocol Π, in the honest majority setting satisfying statistical privacy with knowledge of outputs from Lemma 5.2.
- A statistically secure one-time multi-key MAC scheme $(\mathsf{MK.KeyGen}, \mathsf{MK.Sign}, \mathsf{MK.Verify})$ for multi-bit message space from Corollary 3.4.

Construction. 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. Given any function $F \in \mathrm{NC}^1$, let $F' \in \mathrm{NC}^1$ be an augmented single output functionality that computes F along with a multi-key MAC on the output of F. That is, the function F' takes x_i and $K_i \in \mathcal{K}$ as input from each party P_i and returns $y = F(x_1, \ldots, x_n)$ along with the tag $\tau \leftarrow \mathsf{MK.Sign}(K_1, \ldots, K_n, y)$, where $\mathsf{MK.Sign}$ is the signing algorithm of a one-time multi-key MAC scheme. The protocol proceeds as follows:

- **Rounds 1 and 2:** Each party P_i for $i \in [n]$ samples a key $K_i \leftarrow \mathsf{MK.KeyGen}(1^{\mathbf{k}})$ for the multi-key MAC scheme. It sets its augmented input to $x'_i = (x_i, K_i)$. The parties then execute Π for the augmented functionality F'.
- **Output Evaluation:** Let $y' = (y, \tau)$ denote the output of the protocol Π for the functionality F'. Each party P_i for $i \in [n]$, runs the verification algorithm of the multi-key MAC scheme to verify if τ is a valid multi-key MAC corresponding to y and its key K_i. That is, it checks whether $\mathsf{MK.Verify}(K_i, y, \tau) = 1$. If so, it outputs y, else it outputs $\perp$.

Security. We now give a description of the Ideal world statistical simulator $\mathcal{S}$. $\mathcal{S}$ internally uses the simulator Sim_Π of the underlying protocol Π for functionality F'. Let $\mathcal{A}$ be the adversary. The simulator $\mathcal{S}$ proceeds as follows:

- **Round 1: $\mathcal{S} \to \mathcal{A}$:** Compute the first round simulated messages $\Pi^1_{\mathcal{S}\to\mathcal{A}}$ on behalf of the honest parties using Sim_Π and send it to the adversary.
- **Round 1: $\mathcal{A} \to \mathcal{S}$:** Receive the first round messages $\Pi^1_{\mathcal{A}\to\mathcal{S}}$ from the adversary.
- **Round 2: $\mathcal{S} \to \mathcal{A}$:** Run Sim_Π on inputs $\Pi^1_{\mathcal{A}\to\mathcal{S}}$. When Sim_Π queries the Ideal functionality on extracted inputs $\{x'_i = (x_i, K_i)\}_{i\in\mathcal{A}}$ of the adversary, send $\{x'_i\}_{i\in\mathcal{A}}$ to the Ideal functionality of F and receive y in return. Sample keys for the multi-key MAC scheme on behalf of the honest parties, i.e., for each $i \in [n] \setminus \mathcal{A}$[12], $K_i \leftarrow \mathsf{MK.KeyGen}(1^{\mathbf{k}})$. Use these keys along with the keys extracted from the adversary to compute a multi-key MAC on y, i.e., $\tau \leftarrow \mathsf{MK.Sign}(K_1, \ldots, K_n, y)$. Send (y, τ) to the simulator Sim_Π. Finally Sim_Π outputs the simulated second round messages $\Pi^2_{\mathcal{S}\to\mathcal{A}}$. Send this to the adversary.
- **Round 2: $\mathcal{A} \to \mathcal{S}$:** Receive second round messages $\Pi^2_{\mathcal{A}\to\mathcal{S}}$ from the adversary.
- **Output of Honest Parties:** The simulator reconstructs the output (y^*, τ^*) for the honest parties. It checks whether $(y^*, \tau^*) = (y, \tau)$. If so, it outputs y, else if $\exists i \in [n]\setminus\mathcal{A}$, such that $\mathsf{MK.Verify}(K_i, y^*, \tau^*) = 1$ then output a special abort $\perp^*$, else output $\perp$.

We now prove that the joint distribution of the adversary's view and the output of honest parties is statistically indistinguishable in the Real and Ideal worlds. Given that the underlying protocol Π achieves privacy with knowledge of outputs against malicious adversaries, the view of the adversary simulated by the simulator Sim_Π in the Ideal world is statistically indistinguishable from the view of the adversary in the Real world. Therefore, from the construction of $\mathcal{S}$, it follows that the view of the adversary is statistically indistinguishable in the Real and Ideal worlds. The joint distribution of the view of the adversary and the output of the honest parties is also statistically indistinguishable, conditioned on the simulator not outputting the special abort $\perp^*$. Recall that the simulator outputs $\perp^*$ if $(y', \tau') \neq (y, \tau)$ and $\exists i \in [n] \setminus \mathcal{A}$ such that $\mathsf{MK.Verify}(K_i, y^*, \tau^*) = 1$. Let us assume for the sake of contradiction that the adversary $\mathcal{A}$ can force an output $(y^*, \tau^*) \neq (y, \tau)$ such that $\exists i \in [n] \setminus \mathcal{A}$ such that $\mathsf{MK.Verify}(K_i, y^*, \tau^*) = 1$ with non-negligible probability. We will now construct another adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break the unforgeability of the multi-key MAC scheme with non-negligible probability. The adversary $\mathcal{B}$ interacting with the challenger of the multi-key MAC scheme proceeds as follows:

- **Round 1: $\mathcal{B} \to \mathcal{A}$:** Compute the first round simulated messages $\Pi^1_{\mathcal{B}\to\mathcal{A}}$ on behalf of the honest parties using Sim_Π and send it to the adversary.
- **Round 1: $\mathcal{A} \to \mathcal{B}$:** Receive the first round messages $\Pi^1_{\mathcal{A}\to\mathcal{B}}$ from the adversary.
- **Round 2: $\mathcal{B} \to \mathcal{A}$:** Run Sim_Π on inputs $\Pi^1_{\mathcal{A}\to\mathcal{B}}$. When Sim_Π queries the Ideal functionality on extracted inputs $\{x'_i = (x_i, K_i)\}_{i\in\mathcal{A}}$ of the adversary, send $\{x'_i\}_{i\in\mathcal{A}}$ to the Ideal functionality of F and

[12] We slightly abuse notation here and use $\mathcal{A}$ to also denote the subset of parties controlled by the adversary.

receive y in return. Send the keys $\{K_i\}_{i\in\mathcal{A}}$ extracted from the adversary and the output y to the challenger of the multi-key MAC scheme. The challenger returns a multi-key MAC τ on y. Send (y,τ) to the simulator Sim_Π. Finally Sim_Π outputs simulated second round messages $\Pi^2_{\mathcal{B}\to\mathcal{A}}$. Send this to the adversary.

- **Round 2: $\mathcal{A}\to\mathcal{B}$:** Receive second round messages $\Pi^2_{\mathcal{A}\to\mathcal{B}}$ from the adversary.
- **Output of Honest Parties:** Reconstruct the output (y^*,τ^*) for the honest parties and sends τ^* to the challenger of the multi-key MAC scheme.

Since, $(y^*,\tau^*)\neq(y,\tau)$ and $\exists i\in[n]\setminus\mathcal{A}$ such that $\mathsf{MK.Verify}(K_i,y^*,\tau^*)=1$, adversary $\mathcal{B}$ has managed to break the security of one-time multi-key MACs. But since our multi-key MAC scheme is $\varepsilon(\mathbf{k})$-statistically secure, this can only happen with some negligible probability. Thus, our assumption is incorrect and $\mathcal{A}$ can force such a (y^*,τ^*) only with at most negligible probability. Therefore, overall the joint distribution of the view of the adversary and the output of the honest parties is also statistically indistinguishable.

6 Impossibility of IT-MPC with Public Reconstruction of Output Property

In this section, we show that any IT-MPC protocol that achieves security with abort cannot also satisfy *public reconstruction of output* – a property that we formally define. We prove this by showing implication to information-theoretic one-time signatures, which are known to be impossible [Rom90].

Intuitively, a r-round MPC protocol satisfies the public reconstruction property of output property if: (1) the output of a protocol can be computed given only the broadcast channel messages of all the parties, (2) the broadcast channel messages of all n parties in the first $(r-1)$-rounds and the broadcast channel messages of any $(n-t)$ parties in the the r^{th} round, do not reveal anything about the output of the protocol.

Definition 13 (Public Reconstruction of Output Property)**.** *We say that an r-round MPC protocol Π that achieves security with abort against t malicious corruptions satisfies* public reconstruction of output property *if the following hold:*

1. *There exists a PPT algorithm $\mathsf{Out}_\Pi(.)$ that takes as input the broadcast channel messages of all the parties in the r rounds and computes the output of the protocol.*
2. *There exists a simulator that receives no input (and no access to the ideal functionality) that can simulate a partial transcript consisting of the broadcast channel messages of all parties in the first $(r-1)$ rounds and the broadcast channel messages of any $(n-t)$ parties in the r^{th} round.*

We now proceed to prove the following theorem.

Theorem 6.1. *There does not exist a statistically-secure MPC protocol achieving security with abort, that also satisfies the public reconstruction of output property.*

Proof. Let us assume that there exists an r-round IT-MPC protocol with public reconstruction of output property, that achieves security with abort against t malicious corruptions. We show that the existence of such a protocol Π implies existence of information-theoretic one-time digital signatures, which are known to be impossible.

Let $S=\{1,\ldots,t\}$ be a fixed subset. Let f_0 and f_1 be non-constant (single input) n-party functionalities with boolean outputs. Let Π_0 securely compute f_0 and let Π_1 securely compute f_1. The one-time digital signature scheme can be constructed as follows:

- $(\mathsf{vk},\mathsf{sk})\leftarrow\mathsf{KeyGen}(1^{\mathbf{k}})$: Sample y_0 such that for input vector $\mathbf{x}_0$, where the input of the first party in $\overline{S}$ is y_0 and inputs of all other parties are $\perp$, $f_0(\mathbf{x}_0)=0$. Similarly sample y_1 such that for input vector $\mathbf{x}_1$, where the input of the first party in $\overline{S}$ is y_1 and inputs of all other parties are $\perp$, $f_1(\mathbf{x}_1)=1$. Input

vector $\mathbf{x}_0$ is used in the execution of Π_0 and $\mathbf{x}_1$ is used in execution Π_1. Generate the messages of all the parties in both Π_0 and Π_1, including private channel messages in the first $(r-1)$ rounds. Generate the messages of all the parties in the set $\overline{S}$ in both Π_0 and Π_1 in the r^{th} round.

- **Signing Key:** Let $\mathbf{st}_x^i$ be the private state of party P_i at the end of round $(r-1)$ in Π_x. Set sk to be the private states of all parties in S in both Π_0 and Π_1. That is set $\mathsf{sk} = (\{\mathbf{st}_0^i, \mathbf{st}_1^i\}_{i\in S})$.
- **Verification Key:** Set vk to be the set of broadcast messages of all the parties in the first $(r-1)$ rounds and broadcast messages of parties in S in the last round in both Π_0 and Π_1. That is let $\mathsf{vk}_0 = (\{\Pi_0^{i,j}\}_{i\in S, j\in[r-1]}, \{\Pi_0^{i,j}\}_{i\in\overline{S}, j\in[r]})$ and $\mathsf{vk}_1 = (\{\Pi_1^{i,j}\}_{i\in S, j\in[r-1]}, \{\Pi_1^{i,j}\}_{i\in\overline{S}, j\in[r]})$, where $\Pi_x^{i,j}$ refers to the j^{th} round broadcast channel message of party P_i in Π_x. Set $\mathsf{vk} = (\mathsf{vk}_0, \mathsf{vk}_1)$.

Output $(\mathsf{vk}, \mathsf{sk})$.

- $\sigma \leftarrow \mathsf{Sign}(x, \mathsf{sk})$: Given a message bit $x \in \{0,1\}$, generate the last round broadcast messages of parties in S in the execution of Π_x, i.e., generate $\{\Pi_x^{i,r}\}_{i\in S}$. Output $\sigma = \{\Pi_x^{i,r}\}_{i\in S}$.
- $b \leftarrow \mathsf{Verify}(\mathsf{vk}, x, \sigma)$: Parse $\mathsf{vk} = (\{\Pi_0^{i,j}\}_{i\in S, j\in[r-1]}, \{\Pi_0^{i,j}\}_{i\in\overline{S}, j\in[r]}, \{\Pi_1^{i,j}\}_{i\in S, j\in[r-1]}, \{\Pi_1^{i,j}\}_{i\in\overline{S}, j\in[r]})$ and $\sigma = \{\Pi_x^{i,r}\}_{i\in S}$. Run the public reconstruction algorithm on σ, i.e., run the public reconstruction algorithm on the broadcast messages of all the n parties. Compute

$$x' \leftarrow \mathsf{Out}_\Pi(\{\Pi_x^{i,j}\}_{i\in[n], j\in[r]})$$

If $x' = x$ output 1, else output 0.

Correctness. Correctness follows from the *first condition of the public reconstruction of output* property of the underlying MPC protocol Π.

Unforgeability. We now argue unforgeability of this scheme. For unforgeability, we want to argue that given the verification key vk and a valid signature on x, the adversary should not be able to generate a valid signature on $\overline{x}$. In other words, given $\{\Pi_x^{i,j}\}_{i\in[n], j\in[r]}, \{\Pi_{\overline{x}}^{i,j}\}_{i\in S, j\in[r-1]}, \{\Pi_{\overline{x}}^{i,j}\}_{i\in\overline{S}, j\in[r]}$, no unbounded adversary $\mathcal{A}$ should not be able to compute the $\{\Pi_{\overline{x}}^{i,r}\}_{i\in S}$ such that,

$$\overline{x} \leftarrow \mathsf{Out}_\Pi(\{\Pi_{\overline{x}}^{i,j}\}_{i\in[n], j\in[r]}$$

Sample y' such that for input vector $\mathbf{x}'$, where the input of the first party in $\overline{S}$ is y' and inputs of all other parties are $\perp$, $f_{\overline{x}}(\mathbf{x}') = 0$. Let $\Pi_{\overline{x}'}$ be an execution of Π for functionality $f_{\overline{x}}$, where the input of the first party in $\overline{S}$ is y'. We start by showing that the following two distributions are indistinguishable:

$\mathcal{D}_0$: $\{\{\Pi_x^{i,j}\}_{i\in[n], j\in[r]}, \{\Pi_{\overline{x}}^{i,j}\}_{i\in S, j\in[r-1]}, \{\Pi_{\overline{x}}^{i,j}\}_{i\in\overline{S}, j\in[r]}\}$

$\mathcal{D}_1$: $\{\{\Pi_x^{i,j}\}_{i\in[n], j\in[r]}, \{\Pi_{\overline{x}'}^{i,j}\}_{i\in S, j\in[r-1]}, \{\Pi_{\overline{x}'}^{i,j}\}_{i\in\overline{S}, j\in[r]}\}$

Claim 15. *Let $\boldsymbol{k}$ be the statistical security parameter and $\varepsilon(\boldsymbol{k})$ be a negligible function in $\boldsymbol{k}$. Then the following holds for any distinguisher $\mathcal{A}$:*

$$\mathcal{D}_0 \approx_{s,\varepsilon(\boldsymbol{k})} \mathcal{D}_1$$

Proof. Statistical indistinguishability between distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ follows from the *second condition of the public reconstruction of output* property of Π. □

All that is left to show is that there does not exist any adversary $\mathcal{A}$ that given $\mathcal{D}_1 = \{\{\Pi_x^{i,j}\}_{i\in[n], j\in[r]}, \{\Pi_{\overline{x}'}^{i,j}\}_{i\in S, j\in[r-1]}, \{\Pi_{\overline{x}'}^{i,j}\}_{i\in\overline{S}, j\in[r]}\}$, can compute $\{\Pi_{\overline{x}'}^{i,r}\}_{i\in S}$ such that $\overline{x} \leftarrow \mathsf{Out}_\Pi(\{\Pi_{\overline{x}'}^{i,j}\}_{i\in[n], j\in[r]}$ with an overwhelming probability.

Recall that statistical malicious security with abort guarantees correctness of output of the honest parties, i.e., for Π, we know that for any adversarial strategy, the output of $\Pi_{\overline{x}'}$ is either x or $\perp$ with overwhelming

probability. Let us assume for the sake of contradiction that there exists such an adversary that can compute $\{\Pi^{i,r}_{\overline{x}'}\}_{i\in S}$ such that $\overline{x} \leftarrow \mathsf{Out}_\Pi(\{\Pi^{i,j}_{\overline{x}'}\}_{i\in[n],j\in[r]}$ with overwhelming probability. This is equivalent to a malicious adversary in Π that behaves honestly in the first round and misbehaves in the second round. Since this adversary has managed to force a wrong output on the honest parties, we can now use this adversary to break the malicious security with abort property of the underlying protocol Π. Unforgeability of the proposed signature scheme thus follows. □

7 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.

Theorem 7.1. *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.*

7.1 Construction

We now give a construction of our compiler.

Building Blocks. We use the following ingredients in our construction.

- A two-round MPC Π, in the honest majority setting satisfying statistical 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. We also want the
- 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 Π. 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 Π.

$$(\Pi.\mathsf{msg}_{1,i\to B}, \{\Pi.\mathsf{msg}_{1,i\to j}\}_{j\in[n]}) \coloneqq \Pi_1(1^{\mathbf{k}}, i, x_i; r_i)$$

  where Π_1 is the first round next message function of the protocol Π. $\Pi.\mathsf{msg}_{1,i\to B}$ is the message that is broadcast by P_i in the first round of Π and $\Pi.\mathsf{msg}_{1,i\to j}$ 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 Π as defined in the next round. As defined in Definition 2.2, 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 $(\Pi.\mathsf{msg}_{1,i\to B}, \Pi.\mathsf{msg}_{1,i\to j}, \{K_{i,\ell}^{b,j}\}_{\ell\in[L],b\in\{0,1\}})$ 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:

$$\mathsf{GC}_i \leftarrow \mathsf{Garb}(\mathsf{gk}_i, \Pi_2(1^{\mathbf{k}}, i, x_i, \{\Pi.\mathsf{msg}_{1,j\to i}, \Pi.\mathsf{msg}_{1,i\to j}\}_{j\in[n]}, .; r_i))$$

  where $\Pi_2(1^{\mathbf{k}}, i, x_i, \{\Pi.\mathsf{msg}_{1,j\to i}\}_{j\in[n]}, .; r_i)$ is the second round next message function of party P_i in Π that takes the messages $\{\Pi.\mathsf{msg}_{1,j\to B}\}_{j\in[n]}$ that were broadcast in the first round as input.

- For each $j \in [n] \setminus \{i\}$, it sends $(\mathsf{GC}_i, \{\{K_{j,\ell}^{X_\ell,i}\}_{j\in[n]}\}_{\ell\in[L]})$ to party P_j over a private channel. Here $X = \Pi.\mathsf{msg}_{1,1\to B}||\ldots||\Pi.\mathsf{msg}_{1,n\to B}$ 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.

$$K_{j,\ell}^{X_\ell} := \bigoplus_{i\in[n]} K_{j,\ell}^{X_\ell,i}$$

  and for each $j \in [n]$, it sets

$$\mathbf{K}_j[\Pi.\mathsf{msg}_{1,1\to B}||\ldots||\Pi.\mathsf{msg}_{1,n\to B}] := (K_{j,1}^{X_1}, \ldots, K_{j,1}^{X_L})$$

- For each $j \in [n]$ it evaluates the garbled circuit received in the previous round.

$$\Pi.\mathsf{msg}_{2,j\to B} := \mathsf{Eval}(\mathsf{GC}_j, \mathbf{K}_j[\Pi.\mathsf{msg}_{1,1\to B}||\ldots||\Pi.\mathsf{msg}_{1,n\to B}])$$

- It runs the reconstruction algorithm of Π on $\{\Pi.\mathsf{msg}_{2,j\to B}\}_{j\in[n]}$ to compute the output.

Instantiating Theorem 7.1 using the protocol from Lemma 5.1, we obtain the following corollary:

Corollary 7.2. *There exits a two round MPC over P2P channels for NC^1 functions that achieves statistical security with selective abort against $t < n/2$ malicious corruptions.*

7.2 Proof of Security

We now give a description of the Ideal world simulator $\mathcal{S}$. $\mathcal{S}$ internally uses the simulator Sim_Π of the underlying two-round protocol and the simulator $\mathsf{Sim}_{\mathsf{GC}}$ for the perfectly secure garbling scheme GC. Let $\mathcal{H}$ be the set of honest parties and $\mathcal{A}$ be the adversary. The simulator $\mathcal{S}$ proceeds as follows:

Round 1. $\mathcal{S} \to \mathcal{A}$:

- Simulates the first round messages of P_i for each $i \in \mathcal{H}$ in Π.

$$\{(\Pi.\mathsf{msg}_{1,i\to B}, \{\Pi.\mathsf{msg}_{1,i\to j}\}_{j\in\mathcal{A}})\}_{i\in\mathcal{H}} := \mathsf{Sim}_\Pi(1^{\mathbf{k}})$$

- For each $i \in \mathcal{H}$ simulate

$$(\mathsf{gk}_i, \mathbf{K}_i) \leftarrow \mathsf{Sim}_{\mathsf{GC}}^1(1^{\mathbf{k}}, \varphi(\Pi_2))$$

  where $\varphi(\Pi_2)$ is the topology of the circuit implementing the second round next message function in Π. As defined in Definition 2.2, we parse $\mathbf{K}_i$ as $(K_{i,1}^0, K_{i,1}^1, \ldots, K_{i,L}^0, K_{i,L}^1)$. Here $K_{i,\ell}^0 = K_{i,\ell}^1$ for each $\ell \in [L]$

- Generate random shares $\{\{K_{i,\ell}^{b,j}\}_{j\in\mathcal{A}}\}_{\ell\in[L],b\in\{0,1\}}$ for each $\ell \in [L]$ and $b \in \{0,1\}$
- For each $j \in \mathcal{A}$, it send $(\Pi.\mathsf{msg}_{1,i\to B}, \Pi.\mathsf{msg}_{1,i\to j}, \{K_{i,\ell}^{b,j}\}_{\ell\in[L],b\in\{0,1\}})$ to the adversary over a private channels.

Round 1. $\mathcal{A} \to \mathcal{S}$:

- Receive first round messages of Π, $\{(\Pi.\mathsf{msg}_{1,i\to B}, \{\Pi.\mathsf{msg}_{1,i\to j}\}_{j\in\mathcal{H}})\}_{i\in\mathcal{A}}$ from the adversary.
- Receive shares for wire keys $\{\{K_{i,\ell}^{b,j}\}_{j\in\mathcal{H}}\}_{\ell\in[L],b\in\{0,1\}}$ from the adversary.

Round 2. $\mathcal{S} \to \mathcal{A}$:

- Let $\Pi^i.\mathsf{msg}_{1,j\to B}$ be the value of $\Pi.\mathsf{msg}_{1,j\to B}$ received by the simulator on behalf of party P_i from the the adversarial party P_j. Check if for all $i_1, i_2 \in \mathcal{H}$

$$\{\Pi^{i_1}.\mathsf{msg}_{1,j\to B}\}_{j\in\mathcal{A}} = \{\Pi^{i_2}.\mathsf{msg}_{1,j\to B}\}_{j\in\mathcal{A}}$$

- If the above check goes through:
  - Simulate the second round messages,

$$\{\Pi.\mathsf{msg}_{2,i\to B}\}_{i\in\mathcal{H}} \leftarrow \mathsf{Sim}_\Pi(1^{\mathbf{k}}, \{\Pi.\mathsf{msg}_{1,j\to i}\}_{i\in\mathcal{H},j\in[n]}, \{\Pi.\mathsf{msg}_{1,j\to B}\}_{j\in[n]})$$

  - For each $i \in \mathcal{H}$, simulate a garbled circuit as follows:

$$\mathsf{GC}_i \leftarrow \mathsf{Sim}^2_{\mathsf{GC}}(\mathsf{gk}_i, \mathbf{K}_i, \Pi.\mathsf{msg}_{2,i\to B})$$

    where $\Pi.\mathsf{msg}_{2,i\to B}$ is the simulated second round message party P_i.
  - For each $i \in \mathcal{H}$, choose the remaining shares $\{\{K_{i,\ell}^{X_\ell,j}\}_{j\in\mathcal{H}}\}_{\ell\in[L],b\in\{0,1\}}$ such that for each $\ell \in [L]$,

$$K_{i,\ell}^b \coloneqq \bigoplus_{j\in[n]} K_{i,\ell}^{X_\ell,j}$$

    where $X = \Pi.\mathsf{msg}_{1,1\to B}|| \ldots ||\Pi.\mathsf{msg}_{1,n\to B}$ and X_ℓ denotes the ℓ^{th} bit of X.
  - For each $i \in \mathcal{H}$, $j \in \mathcal{A}$, it sends $(\mathsf{GC}_i, \{\{K_{j,\ell}^{X_\ell,i}\}_{j\in[n]}\}_{\ell\in[L]})$ to the adversarial party P_j over a private channel.
- Else,
  - For each $i \in \mathcal{H}$, simulate a garbled circuit as follows:

$$\mathsf{GC}_i \leftarrow \mathsf{Sim}_{\mathsf{GC}}(\mathsf{gk}_i, 0^L)$$

  - For each $i \in \mathcal{H}$, choose the remaining shares randomly $\{\{K_{i,\ell}^{0,j}\}_{j\in\mathcal{H}}\}_{\ell\in[L]}$.
  - For each $i \in \mathcal{H}$, $j \in \mathcal{A}$, it sends $(\mathsf{GC}_i, \{\{K_{j,\ell}^{0,i}\}_{j\in\mathcal{H}}\}_{\ell\in[L]}, \{\{K_{j,\ell}^{X_\ell^i,i}\}_{j\in\mathcal{A}}\}_{\ell\in[L]})$ to the adversarial party P_j over a private channel.
    Here $X^i = \Pi^i.\mathsf{msg}_{1,1\to B}|| \ldots ||\Pi^i.\mathsf{msg}_{1,n\to B}$ and X_ℓ^i denotes the ℓ^{th} bit of X^i.

Round 2. $\mathcal{A} \to \mathcal{S}$: The simulator receives the second round messages on behalf of the honest parties from the adversary.

We prove security via a sequence of hybrids.

$\mathbf{H}_1$ Execution of the protocol in the real world

$\mathbf{H}_2$ This hybrid is similar to $\mathbf{H}_1$, except that in this hybrid the simulator generates random shares $\{\{K_{i,\ell}^{b,j}\}_{j\in\mathcal{A}}\}_{\ell\in[L],b\in\{0,1\}}$ for each $\ell \in [L]$ and $b \in \{0,1\}$ in the first round. In the second round it checks if for all $i_1, i_2 \in \mathcal{H}$

$$\{\Pi^{i_1}.\mathsf{msg}_{1,j\to B}\}_{j\in\mathcal{A}} = \{\Pi^{i_2}.\mathsf{msg}_{1,j\to B}\}_{j\in\mathcal{A}}$$

where $\Pi^i.\mathsf{msg}_{1,j\to B}$ is the value of $\Pi.\mathsf{msg}_{1,j\to B}$ received by the simulator on behalf of party P_i from the the adversarial party P_j.

- If this check goes through, for each $i \in \mathcal{H}$, it chooses the remaining shares $\{\{K_{i,\ell}^{X_\ell,j}\}_{j\in\mathcal{H}}\}_{\ell\in[L],b\in\{0,1\}}$ such that for each $\ell \in [L]$,

$$K_{i,\ell}^b \coloneqq \bigoplus_{j\in[n]} K_{i,\ell}^{X_\ell,j}$$

  where $X = \Pi.\mathsf{msg}_{1,1\to B}||\ldots||\Pi.\mathsf{msg}_{1,n\to B}$ and X_ℓ denotes the ℓ^{th} bit of X.

- If this check fails, for each $i \in \mathcal{H}$, it chooses the remaining shares randomly $\{\{K_{i,\ell}^{0,j}\}_{j\in\mathcal{H}}\}_{\ell\in[L]}$. For each $i \in \mathcal{H},\ j \in \mathcal{A}$, it sends

  $(\mathsf{GC}_i, \{\{K_{j,\ell}^{0,i}\}_{j\in\mathcal{H}}\}_{\ell\in[L]}, \{\{K_{j,\ell}^{X_\ell^i,i}\}_{j\in\mathcal{A}}\}_{\ell\in[L]})$ to the adversarial party P_j over a private channel. Here $X^i = \Pi^i.\mathsf{msg}_{1,1\to B}||\ldots||\Pi^i.\mathsf{msg}_{1,n\to B}$ and X_ℓ^i denotes the ℓ^{th} bit of X^i.

Claim 16. *Hybrids* $\mathbf{H}_1$ *and* $\mathbf{H}_2$ *are statistically indistinguishable.*

Proof. We rely on the security of garbled circuits to argue indistinguishability of $\mathbf{H}_1$ and $\mathbf{H}_2$. Security of garbled circuits ensures that given a garbled circuit, the probability that the adversary is be able to guess a valid wire key is negligible.

The only difference between hybrid $\mathbf{H}_1$ and $\mathbf{H}_2$ occurs when the adversary sends different values of $\Pi.\mathsf{msg}_{1,j\to B}$ to different honest parties for some $j \in \mathcal{A}$. In this case, in hybrid $\mathbf{H}_1$, the wire keys that it reconstructs correspond to some arbitrary value. While the wire keys that it reconstructs in $\mathbf{H}_2$ correspond to a random value. If there exists an adversary who can distinguish between these hybrids with a noticeable probability, then we can use this adversary to build an adversary who breaks the security of the garbling scheme. □

$\mathbf{H}_3$ This hybrid is similar to hybrid $\mathbf{H}_2$, except that in this hybrid instead of generating the garbled circuits honestly, the simulator simulates them using the simulator of the garbling scheme. More specifically, in the first round, for each $i \in \mathcal{H}$ it computes

$$(\mathsf{gk}_i, \mathbf{K}_i) \leftarrow \mathsf{Sim}_{\mathsf{GC}}^1(1^{\mathbf{k}}, \varphi(\Pi_2))$$

Then in the second round, it checks if for all $i_1, i_2 \in \mathcal{H}$

$$\{\Pi^{i_1}.\mathsf{msg}_{1,j\to B}\}_{j\in\mathcal{A}} = \{\Pi^{i_2}.\mathsf{msg}_{1,j\to B}\}_{j\in\mathcal{A}}$$

where $\Pi^i.\mathsf{msg}_{1,j\to B}$ is the value of $\Pi.\mathsf{msg}_{1,j\to B}$ received by the simulator on behalf of party P_i from the the adversarial party P_j.

- If this check goes through, for each $i \in \mathcal{H}$, it simulates the garbled circuit as follows:

$$\mathsf{GC}_i \leftarrow \mathsf{Sim}_{\mathsf{GC}}^2(\mathsf{gk}_i, \mathbf{K}_i, \Pi.\mathsf{msg}_{2,i\to B})$$

- If this check fails, for each $i \in \mathcal{H}$, it simulates the garbled circuit as follows:

$$\mathsf{GC}_i \leftarrow \mathsf{Sim}_{\mathsf{GC}}^2(\mathsf{gk}_i, \mathbf{K}, 0^L)$$

Claim 17. *From perfect security of the garbling scheme, hybrids* $\mathbf{H}_3$ *and* $\mathbf{H}_4$ *are identical.*

Proof. The indistinguishability of hybrids $\mathbf{H}_3$ and $\mathbf{H}_4$ follows from perfect security of the garbled circuits. The reduction follows from $|\mathcal{H}|(> \frac{n+1}{2})$ invocations of the security of garbled circuits. □

$\mathbf{H}_4$ This hybrid is similar to hybrid $\mathbf{H}_3$, except that in this hybrid instead of computing the messages of Π, using the actual inputs and randomness of the honest parties, it simulates them using the simulator of Π. More specifically, in the first round, it simulates the first round messages of P_i for each $i \in \mathcal{H}$ in Π.

$$\{(\Pi.\mathsf{msg}_{1,i\to B}, \{\Pi.\mathsf{msg}_{1,i\to j}\}_{j\in\mathcal{A}})\}_{i\in\mathcal{H}} \coloneqq \mathsf{Sim}_\Pi(1^\mathbf{k})$$

Then in the second round, it checks if for all $i_1, i_2 \in \mathcal{H}$

$$\{\Pi^{i_1}.\mathsf{msg}_{1,j\to B}\}_{j\in\mathcal{A}} = \{\Pi^{i_2}.\mathsf{msg}_{1,j\to B}\}_{j\in\mathcal{A}}$$

where $\Pi^i.\mathsf{msg}_{1,j\to B}$ is the value of $\Pi.\mathsf{msg}_{1,j\to B}$ received by the simulator on behalf of party P_i from the the adversarial party P_j. If the above check goes through, it simulates the second round messages,

$$\{\Pi.\mathsf{msg}_{2,i\to B}\}_{i\in\mathcal{H}} \leftarrow \mathsf{Sim}_\Pi(1^\mathbf{k}, \{\Pi.\mathsf{msg}_{1,j\to i}\}_{i\in\mathcal{H},j\in[n]}, \{\Pi.\mathsf{msg}_{1,j\to B}\}_{j\in[n]})$$

Claim 18. *From the statistical malicious security of the underlying two-round protocol* Π *for security with abort,* $\mathbf{H}_3$ *and* $\mathbf{H}_4$ *are statistically indistinguishable.*

Proof. The statistical indistinguishability of hybrids $\mathbf{H}_3$ and $\mathbf{H}_4$ follows from the statistical malicious security of the underlying conforming protocol Π. □

Hyb_4 is identical to the ideal world simulator, and hence we're done.

References

[ABT18] Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Perfect secure computation in two rounds. In *Theory of Cryptography - 16th International Conference, TCC 2018*, 2018.

[ABT19] Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Degree 2 is complete for the round-complexity of malicious mpc. 2019. https://eprint.iacr.org/2019/200.

[ACGJ18] Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Round-optimal secure multiparty computation with honest majority. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 395–424, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

[Bea90] Donald Beaver. Multiparty protocols tolerating half faulty processors. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 560–572, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany.

[BGJ+18] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Amit Sahai. Promise zero knowledge and its applications to round optimal MPC. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 459–487, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

[BIB89] Judit Bar-Ilan and Donald Beaver. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In Piotr Rudnicki, editor, *8th ACM Symposium Annual on Principles of Distributed Computing*, pages 201–209, Edmonton, Alberta, Canada, August 14–16, 1989. Association for Computing Machinery.

[BL18] Fabrice Benhamouda and Huijia Lin. k-round mpc from k-round ot via garbled interactive circuits. Technical report, 2018.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd Annual ACM Symposium on Theory of Computing*, pages 503–513, Baltimore, MD, USA, May 14–16, 1990. ACM Press.

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 1–10, Chicago, IL, USA, May 2–4, 1988. ACM Press.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 11–19, Chicago, IL, USA, May 2–4, 1988. ACM Press.

[CD01] Ronald Cramer and Ivan Damgård. Secure distributed linear algebra in a constant number of rounds. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 119–136, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

[Cha90] David Chaum. The spymasters double-agent problem: Multiparty computations secure unconditionally from minorities and cryptographically from majorities. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 591–602, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany.

[CKL03] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 68–86, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany.

[DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 378–394, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany.

[FKN94] Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In *26th Annual ACM Symposium on Theory of Computing*, pages 554–563, Montréal, Québec, Canada, May 23–25, 1994. ACM Press.

[FL82] Michael J. Fischer and Nancy A. Lynch. A lower bound for the time to assure interactive consistency. *Inf. Process. Lett.*, 14(4):183–186, 1982.

[GIKR01] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. The round complexity of verifiable secret sharing and secure multicast. In *33rd Annual ACM Symposium on Theory of Computing*, pages 580–589, Crete, Greece, July 6–8, 2001. ACM Press.

[GIKR02] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. On 2-round secure multiparty computation. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 178–193, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany.

[GIS18] Sanjam Garg, Yuval Ishai, and Akshayaram Srinivasan. Two-round mpc: Information-theoretic and black-box. In *Theory of Cryptography - 16th International Conference, TCC 2018*, 2018.

[GL05] Shafi Goldwasser and Yehuda Lindell. Secure multi-party computation without agreement. *Journal of Cryptology*, 18(3):247–287, July 2005.

[GMS18] Sanjam Garg, Peihan Miao, and Akshayaram Srinivasan. Two-round multiparty secure computation minimizing public key operations. In *Annual International Cryptology Conference*, pages 273–301. Springer, 2018.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[GS17] Sanjam Garg and Akshayaram Srinivasan. Garbled protocols and two-round mpc from bilinear maps. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 588–599. IEEE, 2017.

[GS18a] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 468–499, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[GS18b] Sanjam Garg and Akshayaram Srinivasan. 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*, pages 468–499, 2018.

[HLP11] Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 132–150, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany.

[IK97] Yuval Ishai and Eyal Kushilevitz. 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*, pages 174–184, 1997.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science*, pages 294–304, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press.

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan Eidenbenz, and Ricardo Conejo, editors, *ICALP 2002: 29th International Colloquium on Automata, Languages and Programming*, volume 2380 of *Lecture Notes in Computer Science*, pages 244–256, Malaga, Spain, July 8–13, 2002. Springer, Heidelberg, Germany.

[IK04] Yuval Ishai and Eyal Kushilevitz. On the hardness of information-theoretic multiparty computation. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 439–455, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.

[IKKP15] Yuval Ishai, Ranjit Kumaresan, Eyal Kushilevitz, and Anat Paskin-Cherniavsky. Secure computation with minimal interaction, revisited. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 359–378, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[IKP10] Yuval Ishai, Eyal Kushilevitz, and Anat Paskin. Secure multiparty computation with minimal interaction. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 577–594, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.

[PR18] Arpita Patra and Divya Ravi. On the exact round complexity of secure three-party computation. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 425–458, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

[RBO89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *21st Annual ACM Symposium on Theory of Computing*, pages 73–85, Seattle, WA, USA, May 15–17, 1989. ACM Press.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd Annual ACM Symposium on Theory of Computing*, pages 387–394, Baltimore, MD, USA, May 14–16, 1990. ACM Press.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 162–167. IEEE, 1986.