Round-Optimal Black-Box Secure Computation from Two-Round Malicious OT

Yuval Ishai* Dakshita Khurana† Amit Sahai‡ Akshayaram Srinivasan§

Abstract

We give round-optimal *black-box* constructions of two-party and multiparty protocols in the common random/reference string (CRS) model, with security against malicious adversaries, based on any two-round oblivious transfer (OT) protocol in the same model. Specifically, we obtain two types of results.

1. **Two-party protocol.** We give a (two-round) *two-sided NISC* protocol that makes black-box use of two-round (malicious-secure) OT in the CRS model. In contrast to the standard setting of non-interactive secure computation (NISC), two-sided NISC allows communication from both parties in each round and delivers the output to both parties at the end of the protocol. Prior black-box constructions of two-sided NISC relied on idealized setup assumptions such as OT correlations, or were proven secure in the random oracle model.
2. **Multiparty protocol.** We give a three-round secure multiparty computation protocol for an arbitrary number of parties making black-box use of a two-round OT in the CRS model. The round optimality of this construction follows from a black-box impossibility proof of Applebaum et al. (ITCS 2020). Prior constructions either required the use of random oracles, or were based on two-round malicious-secure OT protocols that satisfied additional security properties.

1 Introduction

The *round complexity* of secure multiparty computation (MPC) has been the subject of intensive research. In this work, we continue this study, focusing on the case of computationally secure MPC protocols without an honest majority. We start with some relevant background.

The semi-honest model. Consider first the simpler setting of *semi-honest* adversaries, who may passively corrupt an arbitrary subset of the parties. In the two-party case, Yao's protocol [Yao86a] is a two-round protocol that can rely on any *two-round oblivious transfer* (OT) protocol. The latter primitive is not only simple and minimal (as a special case of the general result), but also one that pragmatically serves as a useful basis for protocol design. Indeed, two-round OT can be implemented at a low amortized cost with interactive preprocessing [Bea95, IKNP03] and even from scratch [BCG+19]. The question of generalizing Yao's two-round protocol to the *multiparty* setting

*Technion. Email: yuvali@cs.technion.il
†UIUC. Email: dakshita@illinois.edu
‡UCLA. Email: sahai@cs.ucla.edu
§Tata Institute of Fundamental Research. Email: akshayaram.srinivasan@tifr.res.in

remained open for many years. This question was settled by Garg and Srinivasan [GS18] and Benhamouda and Lin [BL18], who showed that two-round OT indeed suffices also for two-round MPC with an arbitrary number of parties.

Black-box vs. non-black-box constructions. A major distinction between Yao's two-party protocol and the recent MPC protocols from [GS18, BL18] is the way in which the OT primitive is used. While the former makes a *black-box*[1] use of OT, in the sense that the construction uses the next-message function of the OT protocol as an oracle, the latter MPC protocols cannot use the OT protocol as an oracle and need to depend on its implementation. This qualitative difference results in a big efficiency gap between the two types of protocols, raising a question about the possibility of a black-box alternative for the multiparty case. Unfortunately, Applebaum et al. [ABG+20] obtained a negative answer: for any $n \geq 3$, general two-round n-party MPC protocols *cannot* make a black-box use of two-round OT. More recently, Patra and Srinivasan [PS21] closed the remaining gap, presenting a black-box construction of *three-round* MPC protocols from two-round OT (improving over a previous four-round protocol from [ACJ17]). This gives us a full understanding of the round complexity of black-box *semi-honest* MPC based on two-round OT.

From semi-honest to malicious. The case of security against malicious adversaries is far less understood. Targeting the goal of matching the round complexity of semi-honest protocols, one needs to rely on a setup assumption. (See Section 1.2 for discussion of results in the plain model.) A minimal form of setup, originating from non-interactive zero knowledge (NIZK) proofs [BFM88], assumes the availability of a *common random string* or, more generally, a (structured) *common reference string*. Our results will apply to both kinds of setup, to which we collectively refer as a *CRS setup*. Given a CRS setup, NIZK can serve as a general round-preserving tool for enforcing an honest behavior by malicious parties. However, general NIZK-based protocols are inherently non-black-box. This raises the following natural question about the round complexity of black-box MPC in the CRS model:

> *Can we make a black-box use of two-round OT to obtain two-round (resp., three-round) two-party (resp., multiparty) MPC protocols with security against malicious adversaries?*

To make this question more precise, we need to specify which kind of two-round OT we consider. Ideally, one would have liked to use *semi-honest* two-round OT as a basis for round-optimal malicious-secure MPC. However, even if we restrict our attention to realizing the OT functionality, it is not known how to construct two-round malicious-secure OT in the CRS model from two-round semi-honest OT, let alone in a black-box way.[2] Indeed, while semi-honest two-round OT protocols are quite easy to construct from essentially every concrete assumption known to imply public-key encryption, obtaining similar protocols with malicious security required ingenious new ideas [PVW08, DGH+20, BF22]. Given this state of affairs, we use *malicious-secure* two-round OT in the CRS model as our basic building block.

[1] A bit more precisely, we refer here to the usual notion of a *fully* black-box reduction [IR90, RTV04], where not only the construction makes a black-box use of OT but also the security reduction makes a black-box use of the adversary.

[2] The same is true even for the plain-model variant of OT with unbounded receiver simulation [NP01, AIR01]. Here we only consider OT and MPC with efficient simulation in the CRS model.

Known results. In the case of two-party "sender-receiver" functionalities, which take inputs from both parties but only deliver the output to the designated receiver, the above question was answered in the affirmative in [IKO+11]. Such two-party protocols, known as *non-interactive secure computation* (NISC) protocols, provide a black-box extension of Yao's protocol that offers security against malicious parties while still requiring only two rounds. However, for general two-party functionalities that deliver outputs to both parties, no analogous result is known. Note that such a "two-sided NISC" protocol cannot be obtained by simply running two instances of standard (one-sided) NISC in parallel, since there is nothing preventing a malicious party from using different inputs in the two executions. The question is similarly open for three-round MPC with $n \geq 3$ parties. Partial progress was made in [PS21], where black-box protocols were constructed from a stronger variant of two-round OT that required some form of adaptive security. However, this extra requirement not only makes the OT primitive qualitatively stronger, but also excludes some existing protocols from the literature (such as CDH- and LPN-based protocols from [DGH+20]). A different kind of progress was recently made in [IKSS21, IKSS22], where positive answers were given using two distinct kinds of idealized setups: either the random oracle model or a random OT correlations setup. To conclude, without strong idealized setups and without strengthening the OT primitive, the above question remained open in both the two-party and the multiparty case.

1.1 Our Contribution

We settle the above question by presenting the first round-optimal black-box constructions of MPC protocols from two-round (malicious-secure) OT in the CRS model. In the two-party case, we obtain the first extension of the black-box NISC protocol from [IKO+11] to general functionalities that deliver outputs to both parties.

Informal Theorem 1. *There is a (two-round, malicious-secure) two-sided NISC protocol in the CRS model that makes a black-box use of two-round malicious-secure OT in the CRS model.*

See Theorem 5.1 for a formal version. From a concrete efficiency perspective, this compiler may be better than the recent random-oracle based compiler from [IKSS22] in that it replaces a computational security parameter by a statistical one.[3] We also obtain an arithmetic variant of this result that makes a black-box use of the underlying field as well as a two-round malicious-secure OLE protocol over the same field [CDI+19, BDM22]. (See Theorem 5.2 for a formal statement.) This variant leverages another advantage of black-box constructions, namely respecting the arithmetic nature of an underlying semi-honest protocol.

Informal Theorem 2. *There is a three-round malicious-secure MPC protocol in the CRS model that makes a black-box use of two-round malicious-secure OT in the CRS model.*

See Theorem 7.1 for a formal statement. The optimality of three rounds follows from the proof of the black-box separation in [ABG+20]. While the main theorem statement of [ABG+20] only refers to a separation between two-round MPC and two-round *semi-honest* OT, the oracle used for the separation actually implies malicious-secure two-round OT.

[3] Jumping ahead, both compilers use a virtual honest-majority MPC protocol in which the number of parties serves as a security parameter. The use of the Fiat-Shamir paradigm in [IKSS22] requires the use of a computational security parameter instead of a statistical one.

Open questions. Our results leave several avenues for future work.

- Is two-round *semi-honest* OT sufficient? As discussed above, the non-triviality of realizing malicious-secure two-round OT from concrete assumptions suggests that even a non-black-box round-preserving compiler from semi-honest to malicious OT would be difficult to obtain. Moreover, the existence of black-box constructions in the random oracle model [MR19, IKSS22] makes a potential black-box separation more challenging.

- Does *three-round* OT suffice in the multiparty case? We conjecture that the black-box separation from [ABG+20] can be extended to rule out this possibility, and leave formalizing this to future work.

- Can similar results be obtained in the *OT-hybrid model*, namely using calls to an ideal OT oracle rather an OT protocol? For standard (one-side) NISC, this is achieved by the construction from [IKO+11]. In the multiparty case, this question is open even in the semi-honest model, where evidence for the difficulty of settling it in the negative is given in [ABG+20]. Our protocols, similarly to the ones from [PS21], inherently make use of the messages generated by the OT protocol, and thus cannot replace the protocol by an oracle.

1.2 Related Work

A long line of work [KOS03, KO04, Wee10, Goy11, ORS15, GMPP16, ACJ17, BHP17, BGJ+18, HHPV18, CCG+20] studied the question of minimizing the round complexity of MPC with security against any number of malicious parties in the *plain model*. In this setting, one cannot hope to match the round complexity of our protocols in the CRS model regardless of the underlying assumptions. Indeed, protocols with black-box simulators must use at least four rounds [KO04, GMPP16]. Even if one allows a non-black-box (polynomial-time) simulation, two-round two-party protocols are unlikely to exist even for the special case of zero knowledge functionalities [BLV03]. We note that there is an interesting line of work [GGJS12, KMO14, GKP17, BGJ+17, BGI+17, ABG+21, FJK21, AMR21] that aims to get around this lower bound by considering a weaker notion of security, namely, security with super-polynomial time simulation (SPS) security.

The quest for minimizing the round complexity of MPC in the plain model, with a standard notion of simulation-based security, culminated in the work of Choudhuri et al. [CCG+20], who obtained a four-round protocol making a *non-black-box* use of four-round malicious-secure OT. The round complexity of this protocol is optimal for protocols with black-box simulation. Additionally requiring the *construction* to be black-box, as in this work, a five-round protocol in the plain model making use of strong flavors of two-round semi-honest OT was given in [IKSS21]. See [CCG+20, IKSS21] and references therein for a more comprehensive survey of this line of work.

2 Technical Overview

In this section, we describe the key technical ideas behind our construction of black-box two-sided NISC (in Section 2.1) and our black-box three-round secure multiparty computation protocol (in Section 2.2).

2.1 Black-Box Two-Sided NISC

Challenges. The main challenge in constructing a two-sided NISC protocol is to design a "black-box" mechanism wherein the adversarial party is somehow forced to use the same input when it plays the role of the sender and the receiver respectively. This is the reason why a natural attempt of constructing a two-sided NISC by running two versions of one-sided NISC in opposite directions fails. The prior works of Ishai et al. [IKSS21, IKSS22] constructed such a "black-box" mechanism by making use of the IPS compiler [IPS08]. However, both these works resorted to idealized setups such as OT correlations, or made use of random oracles in order to implement the watchlist mechanism, one of the key building blocks in this compiler. As we explain later, we encounter significant barriers while trying to remove these idealized setups from the above works. To circumvent these barriers, we develop new techniques to implement this mechanism and prove the security of the protocols based only on a two-message malicious OT. In the rest of the subsection, we elaborate on this in much more detail. As our work also builds on the IPS compiler, we recall the main ideas behind this compiler below.

IPS Compiler. At a high level, the IPS compiler constructs a malicious-secure MPC in the dishonest majority setting via a combination of: (1) a malicious-secure honest-majority client-server MPC (called the outer protocol)[4], and (2) a semi-honest secure protocol in the dishonest majority setting (called the inner protocol). In more detail, each party in the compiled protocol plays the role of a client in the outer protocol. The parties then invoke the inner protocol to emulate the computation done by the servers. Since the outer protocol can be information-theoretic, the computations done by the servers avoid any cryptographic operations. This feature enables the compiled protocol to be black-box. However, as such, this compilation results in an insecure protocol. This is because an adversary can cheat in all the executions of the inner protocol and break their security (as they are only secure against semi-honest adversaries). Since the outer protocol is only guaranteed to be secure as long as a constant ($< 1/2$) fraction of the servers are corrupted, by corrupting all the servers, the adversary has effectively broken the security of the outer protocol and could extract non-trivial information about the honest party inputs. To prevent such an attack, the IPS compiler uses a novel cut-and-choose mechanism referred to as the *watchlist protocol*. In this protocol, each party chooses a random subset of the servers as part of its private "watchlist". The watchlist protocol provides the input and the randomness used by every other party for those server executions that are being watched by this party. The input and randomness of the other server executions are hidden. Every party then checks if the server executions in its watchlist are emulated correctly and aborts if it detects any inconsistency. This guarantees that if the adversary cheats in many server executions, then all the honest parties will detect this and abort, preventing the adversary from learning any useful information about the inputs of the honest parties. On the other hand, if the adversary only cheats in a small number of server executions, then we can rely on the security of the outer MPC protocol to show that the adversary only learns the output of the functionality.

Prior Works. For two-sided NISC, recent black-box protocols from [IKSS21, IKSS22] used the watchlist mechanism to catch a cheating adversary that is *using different inputs* while playing the

[4]By an honest-majority client-server MPC, we mean a setting where a malicious adversary can corrupt any subset of the clients and a constant fraction of the servers.

Citation	Outer Protocol	Inner Protocol	Watchlist Implementation
[IKSS21]	2-round client-server MPC with selective abort	Two-round semi-malicious protocol with first-message equivocality	1-out-of-2 OT correlations model
[IKSS22]	2-round pairwise verifiable MPC	Two-round semi-honest protocol	Random Oracle Model

Table 1: Choice of outer protocol, inner protocol, and idealized model for watchlist implementation in prior works.

role of the sender and the receiver respectively. Specifically, if the adversary is using inconsistent inputs in many server executions, then the honest party detects this via the watchlist mechanism and aborts the execution. On the other hand, if the adversary is using inconsistent inputs in a small number of executions, then the servers that are emulated by these executions can be considered as corrupted in the outer protocol. Since the outer protocol is secure as long as a constant fraction of the servers are corrupted, this prevents the adversary from breaking the privacy of the honest party's inputs. This was the main intuition behind both works. However, these works differed in their choice of the outer protocol, the inner protocol, and the implementation of the watchlist mechanism. We tabulate these choices in Table 1. A common limitation of these two works is their reliance on idealized setups, such as OT correlations [IKSS21] or a random oracle [IKSS22] to implement the watchlist mechanism. We now explain the challenges in trying to remove the idealized setups from these works.

Need for Idealized Models. The key reason why the prior works needed to resort to idealized models is due to a subtle technical difficulty in implementing the IPS compiler. Specifically, the simulator in the IPS compiler needs to know the set of executions that are watched by the corrupted party before it sends its first-round message on behalf of the honest party. Note that in the real world, the honest party's input and randomness corresponding to the adversarial watched executions pass the consistency check and hence, we need to make sure that these checks pass even in the ideal world. Hence, the simulator needs to produce a consistent input and randomness that explains the inner protocol messages in all the executions that are watched by the corrupted party. This is further complicated in the rushing adversarial model where the adversary expects to see the first-round message from the honest party before it sends its own first-round message. Hence, if the set of watched executions are known to the simulator only after it sends the first-round message, then the simulator needs produce randomness that consistently explains the "simulated" first-round inner protocol message w.r.t. some input. In other words, the simulator needs to equivocate the first-round message of the inner protocol. This requires stronger assumptions. However, if we use idealized setups, then the simulator can learn the watched executions of the corrupted party before it sends the first-round message. In particular, this is done by allowing the simulator to implement the dealer while setting up the OT correlations in [IKSS21], or program the output of the random oracle in [IKSS22]. The above issue also precludes a natural attempt of trying to implement the watchlist functionality using a two-round k-out-of-m OT protocol. Indeed, the first-round message that encodes the set of watched executions is sent by the adversary only after it receives the first-round message from the honest party and hence, this approach too requires the first-round message

of the inner protocol to be equivocal.

Our Solution. To overcome this difficulty, we need a watchlist protocol implementation where the simulator *can bias the watched executions* of the corrupted parties whereas the corrupted parties cannot bias the watched executions of the honest parties. These two conflicting features are obtained simultaneously via a *coin-tossing protocol*. Specifically, the watched executions of each party is sampled randomly where the randomness is contributed by both the parties and not just by the receiver party. This ensures that the simulator can set the randomness on behalf of the honest party in such a way that the corrupted party receives a randomly sampled set of executions that was chosen prior to sending the first-round message. At the same time, since the receiver party also provides a part of the randomness, this ensures that the corrupted party cannot bias the set of watched executions of the honest party. This helps in overcoming the above mentioned technical difficulty in implementing the IPS compiler. The next question is can we construct such a watchlist protocol? Indeed, the work of Ishai et al. [IKO+11] provides an instantiation that makes black-box use of a two-round malicious-secure OT. However, such a watchlist protocol alone does not solve all the issues and we elaborate more on this below.

Need for Watchlist Output at the End of the First Round. While the above watchlist protocol ensures that the simulator has the power to bias the watched executions of the corrupted parties, it leads to new incompatibility issues with the prior techniques. Specifically, the prior works crucially relied on the output of the watchlist protocol to be delivered to the honest parties at the end of the first round. This is indeed possible if we rely on idealized setups. However, in the above described approach, the honest party learns the output of the watchlist protocol only after the corrupted party sends its second-round message. Hence, it can only perform all the watchlist checks after it has sent its final round message in the protocol (since we are dealing with rushing adversaries). This leads to new problems and let us explain them in a bit more detail.

Firstly, the work of Ishai et al. [IKSS21] considered a two-round semi-honest inner protocol where the first-round message could be equivocated (see Table 1). However, a malicious party can also equivocate its first-round message and thereby, break the security of the inner protocol. This was not a problem in their setting since the output of the watchlist protocol is made available to the honest parties at the end of the first round (this is possible in the OT correlations model). Hence, the honest party can detect if the first-round message is equivocated in many inner protocol executions and abort if it is the case. However, in our watchlist protocol, the output is delivered to the honest party only at the end of the second round. By this time, the honest party would have sent the second-round message in the inner protocol and the adversary could potentially recover the entire input of the uncorrupted party.

In a more recent work, Ishai et al. [IKSS22] removed the need for an inner protocol with first message equivocality by considering a "stronger" outer protocol. This outer protocol which they termed as pairwise verifiable MPC protocol is a two-round client-server MPC protocol that additionally satisfies a special error correction property. Specifically, for any choice of second-round message from the corrupted servers, the error correction property requires that the output of the honest client remains the same. Unfortunately, obtaining such a protocol against standard malicious adversaries is hard due to the known barriers [GIKR02]. To overcome this, Ishai et al. considered security against weaker adversaries called pairwise verifiable adversaries. Roughly speaking, pairwise verifiable adversarial clients are restricted to send a first-round message such

that the messages received by all the honest servers pass some consistency check. However, this restriction of only considering pairwise verifiable adversaries also seems incompatible with our watchlist protocol. Specifically, before we send the second-round message, we need to make sure that the first-round message in the outer protocol pass the pairwise consistency check and we must proceed only if these checks pass. In the work of Ishai et al. [IKSS22], this was made possible by making use of a random oracle. But in our setting, since the output of the watchlist functionality is only delivered after the honest party sends the second-round message, we cannot perform this check before sending the final message. Thus, an adversarial party can use first-round messages in the outer protocol that do not pass the pairwise consistency check and completely break the privacy of the honest party's inputs.

Our Approach. We note that the above mentioned incompatibility issue could be alleviated if we use a two-round malicious secure oblivious transfer protocol that has equivocal first-round message [GS18, PS21]. Specifically, such an OT protocol forces a corrupt receiver to send a valid first-round message but enables the simulator to equivocate the first-round message to both bits 0 and 1. Indeed, a malicious party is forced to send a valid first-round message whereas the simulator could equivocate the first-round message as in [IKSS21]. However, we do not know of a black-box construction of this primitive from any two-round malicious secure OT protocol.[5] Moreover, recent protocols from the literature (such as ones based on CDH and LPN [DGH+20]) do not satisfy this property. Our goal here is to overcome this issue by only making black-box use of a two-round malicious secure OT.

Instead of relying on a pairwise verifiable MPC protocol, our solution to this problem is to rely on a standard outer protocol satisfying security with abort, say for instance, the one given by Ishai, Kushilevitz, and Paskin [IKP10, Pas12]. To make this outer protocol compatible with the IPS compiler, Ishai et al. [IKSS21] observed that the inner protocol needed to additionally satisfy first-round equivocality (see Table 1). The key insight behind our solution is that first-round equivocality of the inner protocol is actually on overkill and we could instead use a far weaker security property. We now explain this in detail.

Recall that the watchlist mechanism is guaranteed to catch a malicious party that cheats in a large number of inner protocol executions. However, a malicious party can cheat in a small number of executions such that it goes undetected by the watchlist of the honest party with some non-negligible probability. In this case, we should be able to rely on the security of the outer protocol as the number of malicious server corruptions is "small". However, in order to invoke this security property, we need to compute the inner protocol output received by the honest party in each of the executions where the adversary has cheated. This corresponds to the second-round message sent by the corrupted servers to the honest client and we need to provide this information to the simulator of the outer protocol. [IKSS21] argued that if the inner protocol satisfies first-round equivocality then it is possible for the simulator to compute this output. In particular, the simulator can equivocate the first-round message as per the honest party's input and then use the corresponding randomness to compute the output of this inner protocol execution. In this work, we observe that this property can be weakened, specifically, to what we call as *output equivocality*. This property requires that if the adversary cheated in generating the second-round message, then the simulator (that is additionally provided the input of the honest party) must produce an output that is computationally indistinguishable from the honest party's output in the real execution.

[5] We note that [GS18] gave a non-black-box construction.

Specifically, instead of requiring the entire first-round message to be equivocable, we only need the output computation to be equivocable. This property is implied by first message equivocality and could be potentially be realized under weaker assumptions. Further, since the output of our watchlist protocol is only delivered after we send the second-round message, we need our inner protocol to also be secure against malicious receivers. Hence, it is sufficient to construct an inner protocol that is secure against malicious receivers and also satisfies output equivocality.

Somewhat surprisingly, both of these properties can be obtained simultaneously if we simply replace the two-round semi-honest OT in the Yao's protocol with a two-round malicious secure OT. Specifically, the security against malicious receivers follows from the folklore observation about Yao's protocol when instantiated with a two-round malicious secure OT protocol. The output equivocality property is argued using the security of the oblivious transfer against malicious senders. In particular, the simulator could use the extractor for the OT protocol and extract the set of both labels for each input wire of the garbled circuit that was generated. Now, given the honest party's input, the output equivocal simulator can just evaluate the received garbled circuit on the chosen set of labels according to the honest receiver's input and output the result of the evaluation. From the sender security of the OT protocol, we infer that the output of this evaluation is computationally indistinguishable from the honest evaluation. This allows us to construct an inner protocol with the desired properties and thereby instantiate the IPS compiler.

The full description of the inner protocol along with the security properties it needs to satisfy is given in Section 4. The construction and the security analysis of our two-sided black-box NISC protocol can be found in Section 5.

Further Remarks. We observe that there is no need to rely on a special inner protocol that was constructed based on Yao's garbled circuits. Instead, we can start with any one-sided OT-based NISC protocol. This follows from the fact that security against malicious receivers comes for free, and the output equivocality follows from security of the one-sided NISC against malicious senders. Thus, our work can be viewed as a black-box construction of two-sided NISC from any one-sided NISC. This allows us to directly transfer any efficiency improvements in the one-sided NISC setting to the more challenging two-sided NISC. Furthermore, this allows us to upgrade known one-sided NISC protocols in the arithmetic setting [CDI$^+$19, DIO21] (making a black-box use of the underlying field) to similar two-sided NISC protocols.

2.2 Black-Box Three-Round MPC

To construct a black-box three-round MPC protocol, we again rely on the IPS compiler. Specifically, we start with an outer protocol that supports an arbitrary number of clients and satisfying security with selective abort (such a protocol was constructed in [IKP10, Pas12]). As in the black-box two-sided NISC case, we implement the watchlist protocol via a coin-tossing based approach. This enables the simulator to bias the watched executions of the corrupted parties before it sends its first-round message on behalf of the honest parties. The only difference from the two-sided NISC case is that we need to rely on an inner protocol that runs in three rounds (due to the black-box impossiblity of [ABG$^+$20]). To make the inner protocol compatible with the above outer protocol, we need it to satisfy the following two additional properties:

- **Robustness:** Even if the adversary cheats in generating the messages in the first two rounds of the protocol, it cannot break the privacy of the honest party inputs. This is needed since

the output of the watchlist is delivered only at the end of the second round and any cheating in the first two rounds should not enable the corrupted party to break the privacy of the honest parties.

- **Last Round Equivocality:** If the adversary has cheated in the first two rounds, then the simulator when provided with the inputs of all the honest parties must produce a last round message which is computationally indistinguishable from the real execution. This is needed to generate the last round message in the inner protocol executions where the adversary has cheated in the first two rounds.

We note that robustness and last round equivocality was also needed in the inner protocol used in [IKSS21]. However, their inner protocols could either run in two rounds (in the presence of OT correlations), or four rounds in the plain model. Here, our focus is on constructing such an inner protocol in three rounds in the CRS model.

Constructing Multiparty Inner Protocol. Our first observation is that to construct such an inner protocol for computing arbitrary functionalities, it is sufficient to construct an inner protocol that computes the 3MULTPlus functionality. 3MULTPlus is a special multiparty functionality that takes (x_1, y_1) from the first party, (x_2, y_2) from the second party, and (x_3, y_3) from the third party where x_i, y_i are bits and delivers $x_1 \cdot x_2 \cdot x_3 + y_1 + y_2 + y_3$ to all the parties. Indeed, the standard bootstrapping results from 3MULTPlus to general functions [BGI$^+$18, GIS18, ABG$^+$20] for the case of semi-honest adversaries also extends to the above security definition. Thus, it is sufficient to construct an inner protocol for 3MULTPlus functionality that satisfies both robustness and last round equivocality.

The starting point of our construction of such a protocol is the work of Patra and Srinivasan [PS21] who gave a construction in the semi-honest setting based on any two-round semi-honest OT protocol. The main result that we prove is that if we replace the two-round semi-honest OT protocol in their construction with a two-round malicious-secure version, then the resultant protocol is robust. However, proving this is not straightforward and requires a careful security analysis (this appears in Proposition 6.2). To prove last round equivocality, we observe that the last round message sent by each party in the protocol of [PS21] is obtained by decrypting some sender OT message. As in the case of two-sided NISC setting, we show that this message can be equivocated if the two-round OT protocol is secure against malicious senders. This allows us to construct a three-round inner protocol that satisfies robustness and equivocality by making black-box use of a two-round malicious-secure OT.

The formal description of the security properties along with the construction and the proof of security of the multiparty inner protocol appears in Section 6.

Putting things together. As mentioned before, our three-round black-box multiparty protocol is obtained by combining the two-round coin-tossing based watchlist protocol along with a three-round inner protocol satisfying both robustness and last round equivocality. At the end of the second round, the output of the watchlist protocol is delivered to all the parties. If the adversary cheats in many inner protocol executions, then this is detected by the honest parties who abort before sending the final round message. In this case, we rely on the robustness property of the inner protocol to show that the adversary learns no information about the private inputs of the honest parties. On the other hand, if the adversary only cheats in a small number of executions, then

we corrupt the corresponding servers in the outer protocol. We use the last round equivocality to generate the final message in the inner protocol for these executions. We finally rely on the security of the outer protocol to argue that only the output of the functionality is leaked to the adversary since the number of server corruptions is "small".

The construction of the three-round black-box MPC protocol and the proof of security can be found in Section 7.

2.3 Another Perspective

A different way to view our techniques is as follows. Let us start with the simplest, round-optimal semi-honest protocols for 2PC and MPC that make black-box use of two-round semi-honest OT. For the case of two parties, we consider Yao's protocol and for the case of multiple parties, we consider the protocol from the work of Patra and Srinivasan [PS21]. In both these protocols, we replace the underlying semi-honest OT protocol with a malicious secure OT protocol and ask what security properties are satisfied by this modification. In this work, we show that the properties satisfied correspond to that of the inner protocols. Later, we use the IPS compiler to bootstrap this "weaker" security notion to the standard malicious security. However, this runs into several technical hurdles (as explained earlier) and we develop new techniques to overcome them.

3 Preliminaries

Let λ denote the cryptographic security parameter. We assume that all cryptographic algorithms implicitly take 1^λ as input. A function $\mu(\cdot) : \mathbb{N} \to \mathbb{R}^+$ is said to be negligible if for any polynomial $\mathsf{poly}(\cdot)$, there exists λ_0 such that for all $\lambda > \lambda_0$, we have $\mu(\lambda) < \frac{1}{\mathsf{poly}(\lambda)}$. We will use $\mathsf{negl}(\cdot)$ to denote an unspecified negligible function and $\mathsf{poly}(\cdot)$ to denote an unspecified polynomial function. For any $i \in [n]$, let x_i denote the symbol at the i-th co-ordinate of x, and for any $T \subseteq [n]$, let $x_T \in \{0,1\}^{|T|}$ denote the projection of x to the co-ordinates indexed by T. We use $\mathsf{supp}(X)$ to denote the support of a random variable X.

For a probabilistic algorithm A, we denote $A(x; r)$ to be the output of A on input x with the content of the random tape being r. When r is omitted, $A(x)$ denotes a distribution. For a finite set S, we denote $x \leftarrow S$ as the process of sampling x uniformly from the set S. We will use PPT to denote Probabilistic Polynomial Time algorithm. We assume w.l.o.g. that the length of the randomness for all cryptographic algorithms is λ.

We say that two distribution ensembles $\{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $\{Y_\lambda\}_{\lambda\in\mathbb{N}}$ are computationally indistinguishable if for every non-uniform PPT distinguisher D there exists a negligible function $\mathsf{negl}(\cdot)$ such that $|\Pr[D(1^\lambda, X_\lambda) = 1]| - \Pr[D(1^\lambda, Y_\lambda) = 1]| \leq \mathsf{negl}(\lambda)$.

3.1 Two-Round Malicious-Secure Oblivious Transfer

For self-containment, we include in Appendix C the standard definitions of secure multiparty computation to which our main results apply.

Here we formally define two-round OT in the CRS model, which our protocols use as a black box. We follow the previous definitions of two-round OT from the literature [PVW08, DGH$^+$20], which in turn are a simple instance of general UC security definitions [Can20, CLOS02].[6] Note

[6] One further simplification we make is eliminating the auxiliary information given by the environment to both the

that since this definition respects parallel composition, one can realize multiple parallel OT calls by making a black-box use of such a (single) OT protocol.

In more detail, a two-round malicious secure oblivious transfer protocol is given by a tuple of PPT algorithms $(\mathsf{CRSGen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{out}_{\mathsf{OT}})$. The CRSGen algorithm takes in the security parameter 1^λ (encoded in unary) and outputs the common reference string crs. OT_1 algorithm is run by the receiver and takes in crs and the input b of the receiver, and outputs otr. OT_2 algorithm is run by the sender of the OT protocol and takes in crs, otr and the sender inputs (m_0, m_1), and outputs ots. The $\mathsf{out}_{\mathsf{OT}}$ algorithm takes in $\mathsf{crs}, \mathsf{ots}, b$ and the random tape used in generating otr and outputs m_b.

Definition 3.1. *A two-round oblivious transfer protocol* $(\mathsf{CRSGen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{out}_{\mathsf{OT}})$ *is said to be secure against malicious adversaries if there exists a (stateful) PPT simulator-extractor pair* $(\mathsf{Sim}_{\mathsf{OT}}, \mathsf{Ext}_{\mathsf{OT}})$ *such that the following properties hold:*

- ***Security against Malicious Receiver:*** *For any (stateful) non-uniform PPT adversary* $\mathcal{A}$ *corrupting the receiver, we have:*

$$\mathsf{Real}_R(1^\lambda, \mathcal{A}) \approx_c \mathsf{Ideal}_R(1^\lambda, \mathcal{A}, (\mathsf{Sim}_{\mathsf{OT}}, \mathsf{Ext}_{\mathsf{OT}}))$$

  where the Real_R *and* Ideal_R *experiments are described in Figure 1.*

- ***Security against Malicious Sender:*** *For any (stateful) non-uniform PPT adversary* $\mathcal{A}$ *corrupting the sender and for any receiver input* b*, we have:*

$$\mathsf{Real}_S(1^\lambda, \mathcal{A}, b) \approx_c \mathsf{Ideal}_S(1^\lambda, \mathcal{A}, b, (\mathsf{Sim}_{\mathsf{OT}}, \mathsf{Ext}_{\mathsf{OT}}))$$

  where the Real_S *and* Ideal_S *experiments are described in Figure 2.*

$\mathsf{Real}_R(1^\lambda, \mathcal{A}, (m_0, m_1))$	$\mathsf{Ideal}_R(1^\lambda, \mathcal{A}, (m_0, m_1), (\mathsf{Sim}_{\mathsf{OT}}, \mathsf{Ext}_{\mathsf{OT}}))$
1. $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda)$.	1. $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Sim}_{\mathsf{OT}}(1^\lambda, R)$.
2. $\mathsf{otr}, (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{crs})$.	2. $\mathsf{otr}, (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{crs})$.
3. $\mathsf{ots} \leftarrow \mathsf{OT}_2(\mathsf{crs}, \mathsf{otr}, (m_0, m_1))$.	3. $b \leftarrow \mathsf{Ext}_{\mathsf{OT}}(R, \mathsf{td}, \mathsf{otr})$.
4. Output $\mathcal{A}(\mathsf{crs}, \mathsf{ots})$.	4. $\mathsf{ots} \leftarrow \mathsf{OT}_2(\mathsf{crs}, \mathsf{otr}, (m_b, m_b))$.
	5. Output $\mathcal{A}(\mathsf{crs}, \mathsf{ots})$.

Figure 1: Descriptions of Real_R and Ideal_R experiments.

Remark 3.2. *The above definition of oblivious transfer protocol, which we use as our main primitive, is satisfied by any two-round UC-secure protocol realizing the ideal OT functionality in the CRS model. In our general security definitions (see Appendix C) we only consider standalone security for simplicity, though our protocols can actually be shown to be UC-secure in the CRS model when using UC-secure 2-round OT.*

adversary and the simulator. Instead, our definitions allow both the adversary and the simulator to be independently nonuniform, which results in a slightly weaker security notion. However, all of our protocols also satisfy the standard security notion from the literature.

$\mathsf{Real}_S(1^\lambda, \mathcal{A}, b)$

1. $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda)$.
2. $\mathsf{otr} \leftarrow \mathsf{OT}_1(\mathsf{crs}, b; r)$ where $r \leftarrow \{0,1\}^\lambda$.
3. $\mathsf{ots} \leftarrow \mathcal{A}(\mathsf{crs}, \mathsf{otr})$.
4. Output $\mathcal{A}(\mathsf{crs}, \mathsf{otr}, \mathsf{out}_{\mathsf{OT}}(\mathsf{crs}, \mathsf{ots}, (b, r)))$.

$\mathsf{Ideal}_R(1^\lambda, \mathcal{A}, (m_0, m_1), (\mathsf{Sim}_{\mathsf{OT}}, \mathsf{Ext}_{\mathsf{OT}}))$

1. $(\mathsf{crs}, \mathsf{td}, \mathsf{otr}) \leftarrow \mathsf{Sim}_{\mathsf{OT}}(1^\lambda, S)$.
2. $\mathsf{ots} \leftarrow \mathcal{A}(\mathsf{crs}, \mathsf{otr})$.
3. $(m_0, m_1) \leftarrow \mathsf{Ext}_{\mathsf{OT}}(S, \mathsf{td}, \mathsf{ots})$.
4. Output $\mathcal{A}(\mathsf{crs}, \mathsf{otr}, m_b)$.

Figure 2: Descriptions of Real_S and Ideal_S experiments.

3.2 Two-Round Client-Server Protocol

A two-round client-server protocol between n clients and m servers for computing a function f is given by a tuple of algorithms $(\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$. Share is a PPT algorithm takes in 1^λ, the identity i of the client (which is an index in $[n]$), its private input x_i and outputs $(x_{i,1}, \ldots, x_{i,m})$. Eval algorithm takes in the identity j of the server (which is an index in $[m]$), $(x_{1,j}, \ldots, x_{n,j})$ and outputs $(z_j^1, \ldots, z_j^n)$. Dec algorithm takes in the identity i of the client, $(z_1^i, \ldots, z_m^i)$, its private random tape and outputs z.

Let $\mathcal{A}$ be a malicious adversary that is corrupting an arbitrary subset M of the clients and a set S of the servers. Let H denote the set of uncorrupted clients. In the following, we use $\mathsf{Real}(1^\lambda, \Pi, \mathcal{A}, \{x_i\}_{i \in H})$ to denote the joint distribution of the output of the adversary and the output of the honest clients (when using input x_i for each $i \in H$) in the real execution of the protocol.

Let Sim be a malicious adversary that is corrupting the subset M of the clients in the ideal protocol execution. Recall that in the ideal execution, the parties have access to a trusted functionality for computing f. The honest parties send their inputs to this trusted functionality, whereas the adversarial party can send an arbitrary value. The functionality computes the output and sends it to Sim. For each of the honest clients, Sim sends the instruction to the trusted party whether to deliver this output to this honest client or instruct it to abort. For each $i \in H$ to which the ideal functionality delivers the output, the honest client i outputs whatever is received from the trusted functionality. The rest of the honest clients output $\perp$. We denote $\mathsf{Ideal}(1^\lambda, \Pi, \mathsf{Sim}, \{x_i\}_{i \in H})$ to denote the joint distribution of the output of Sim and the output of the honest clients when their inputs are $\{x_i\}_{i \in H}$ in the ideal execution of the protocol.

Definition 3.3. *A two-round n-client m-server protocol* $(\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ *is said to compute a function f with t server corruptions with selective abort if for any non-uniform PPT real world adversary $\mathcal{A}$ that is corrupting an arbitrary subset M of the clients and a subset S of the servers where $|S| \leq t$, there exists a non-uniform PPT ideal world adversary* Sim *corrupting the subset M such that for any inputs $\{x_i\}_{i \in [n] \setminus M}$ of the uncorrupted parties, we have:*

$$\mathsf{Real}(1^\lambda, \Pi, \mathcal{A}, \{x_i\}_{i \in [n] \setminus M}) \approx_c \mathsf{Ideal}(1^\lambda, \Pi, \mathsf{Sim}, \{x_i\}_{i \in [n] \setminus M})$$

Privacy with Knowledge of Outputs. We also consider a weaker version of security termed as privacy with knowledge of outputs which was introduced in [IKP10]. In this weaker model, the ideal world adversary Sim is allowed to set the outputs of the honest parties. This is modeled by

allowing Sim to send the outputs of the honest parties to the trusted functionality which is then delivered to them.

3.3 Garbled Circuits

We recall the definition of garbling scheme for circuits [Yao86b] (see Applebaum et al. [AIK05, App17], Lindell and Pinkas [LP09] and Bellare et al. [BHR12] for a detailed proof and further discussion). A garbling scheme for circuits is a tuple of PPT algorithms $(\mathsf{Garble}, \mathsf{Eval})$. Garble is the circuit garbling procedure and Eval is the corresponding evaluation procedure. More formally:

- $(\widetilde{C}, \{\mathsf{lab}_{i,b}\}_{i\in[n],b\in\{0,1\}}) \leftarrow \mathsf{Garble}\left(1^\lambda, C\right)$: Garble takes as input a security parameter 1^λ, a circuit $C : \{0,1\}^n \to \{0,1\}^m$, and outputs a *garbled circuit* $\widetilde{C}$ along with *labels* $\mathsf{lab}_{i,b}$ where $i \in [n]$ and $b \in \{0,1\}$. Each label $\mathsf{lab}_{i,b}$ is assumed to be in $\{0,1\}^\lambda$.
- $y \leftarrow \mathsf{Eval}\left(\widetilde{C}, \{\mathsf{lab}_{i,x_i}\}_{i\in[n]}\right)$: Given a garbled circuit $\widetilde{C}$ and a sequence of input labels $\{\mathsf{lab}_{i,x_i}\}_{i\in[n]}$ where $x \in \{0,1\}^n$ (referred to as the garbled input), Eval outputs a string y.

Correctness. For correctness, we require that for any circuit $C : \{0,1\}^n \to \{0,1\}^m$ and any input $x \in \{0,1\}^n$, we have that:

$$\Pr\left[C(x) = \mathsf{Eval}\left(\widetilde{C}, \{\mathsf{lab}_{i,x_i}\}_{i\in[n]}\right)\right] = 1$$

where $(\widetilde{C}, \{\mathsf{lab}_{i,b}\}_{i\in[n],b\in\{0,1\}}) \leftarrow \mathsf{Garble}\left(1^\lambda, C\right)$.

Security. For security, we require that there exists a PPT simulator $\mathsf{Sim}_{\mathsf{GC}}$ such that for any circuit $C : \{0,1\}^n \to \{0,1\}^m$ and any input $x \in \{0,1\}^n$, we have that:

$$\left(\widetilde{C}, \{\mathsf{lab}_{i,x_i}\}_{i\in[n]}\right) \approx_c \mathsf{Sim}_{\mathsf{GC}}\left(1^\lambda, 1^{|C|}, 1^{|x|}, C(x)\right)$$

where $(\widetilde{C}, \{\mathsf{lab}_{i,b}\}_{i\in[n],b\in\{0,1\}}) \leftarrow \mathsf{Garble}\left(1^\lambda, C\right)$.

4 Two-Party Inner Protocol

In this section, we give a definition of a two-party protocol that satisfies some special properties (known as two-party inner protocol). We give a construction of such a two-party inner protocol making black-box use of a two-round malicious-secure OT. In the next section, we use this protocol to construct a two-sided NISC.

4.1 Definition

A two-round two-party protocol for computing a two-party function f is given by a tuple of PPT algorithms $(\mathsf{Setup}, \Pi_1, \Pi_2, \mathsf{out}_\Pi)$. Setup algorithm takes in the security parameter 1^λ (encoded in unary) and outputs the common reference string crs. Π_1 is run by the receiver and takes in crs and the receiver input x_0 and outputs π_1. Π_2 is run by the sender and takes in crs, π_1, the sender input x_1 and outputs π_2. out_Π takes in crs, π_2, x_0 and the random tape of Π_1 and outputs $f(x_0, x_1)$.

Definition 4.1. *A two-party protocol* $(\mathsf{Setup}, \Pi_1, \Pi_2, \mathsf{out}_\Pi)$ *for computing a functionality* f *that delivers the output to the receiver is said to be a two-party inner protocol if there exists a (stateful) PPT simulator-extractor pair* $(\mathsf{Sim}_\Pi, \mathsf{Ext}_\Pi)$ *such that the following properties hold:*

- ***Security Against Malicious Receivers:*** *For any (stateful) non-uniform PPT adversary* $\mathcal{A}$ *corrupting the receiver and for any sender input* x_1*, we have:*

$$\mathsf{Real}_R(1^\lambda, \mathcal{A}, x_1) \approx_c \mathsf{Ideal}_R(1^\lambda, \mathcal{A}, x_1, (\mathsf{Sim}_\Pi, \mathsf{Ext}_\Pi))$$

  where Real_R *and* Ideal_R *are described in Figure 3.*

- ***Correctness of Extraction.*** *For any non-uniform PPT adversary* $\mathcal{A}$ *corrupting the receiver, we have*

$$\Pr\Big[\mathsf{Ext}_\Pi(R, \mathsf{td}, \Pi_1(\mathsf{crs}, x_0; r_0)) \neq x_0 \Big| \\ (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Sim}_\Pi(1^\lambda, R), (x_0, r_0) \leftarrow \mathcal{A}(\mathsf{crs})\Big] \leq \mathsf{negl}(\lambda)$$

- ***Robust Security Against Semi-Malicious Senders (a.k.a., output equivocality):*** *Informally, this property requires that if a malicious sender sends a second round message that is not explainable (by providing a valid (input, randomness) pair), then we require an equivocal simulator that when given the private input of the honest receiver computes an output such that the joint distribution of the view of* $\mathcal{A}$ *and the output of the honest receiver in the real execution is indistinguishable to the ideal execution using this special simulator. Formally, for any (stateful) non-uniform PPT adversary* $\mathcal{A}$ *corrupting the sender and for any receiver input* x_0*, we have:*

$$\mathsf{Real}_S(1^\lambda, \mathcal{A}, x_1) \approx_c \mathsf{Ideal}_S(1^\lambda, \mathcal{A}, x_0, (\mathsf{Sim}_\Pi, \mathsf{Ext}_\Pi))$$

  where Real_S *and* Ideal_S *are described in Figure 4.*

$\mathsf{Real}_R(1^\lambda, \mathcal{A}, x_1)$	$\mathsf{Ideal}_R(1^\lambda, \mathcal{A}, x_1, (\mathsf{Sim}_\Pi, \mathsf{Ext}_\Pi))$
1. $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$.	1. $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Sim}_\Pi(1^\lambda, R)$.
2. $\pi_1 \leftarrow \mathcal{A}(\mathsf{crs})$.	2. $\pi_1 \leftarrow \mathcal{A}(\mathsf{crs})$.
3. $\pi_2 \leftarrow \Pi_2(\mathsf{crs}, \pi_1, x_1)$.	3. $x_0 \leftarrow \mathsf{Ext}_\Pi(R, \mathsf{td}, \pi_1)$.
4. Output $\mathcal{A}(\pi_2)$.	4. $\pi_2 \leftarrow \mathsf{Sim}(\mathsf{crs}, x_0, f(x_0, x_1))$.
	5. Output $\mathcal{A}(\pi_2)$.

Figure 3: Descriptions of Real_R and Ideal_R experiments.

Remark 4.2. *We note that correctness of extraction is implicitly implied by security against malicious receivers. However, for the ease of usage in the next section, we state it as a separate property.*

In this Appendix A, we prove the following proposition.

$\mathsf{Real}_S(1^\lambda, \mathcal{A}, x_0)$

1. $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $\pi_1 \leftarrow \Pi_1(\mathsf{crs}, x_0; r_0)$ where $r_0 \leftarrow \{0,1\}^\lambda$.
3. $(\pi_2, (x_1, r_1)) \leftarrow \mathcal{A}(\mathsf{crs}, \pi_1)$.
4. Output $(\mathsf{crs}, \pi_1, \mathsf{out}_\Pi(\mathsf{crs}, \pi_2, (x_0, r_0)))$.

$\mathsf{Ideal}_S(1^\lambda, \mathcal{A}, x_0, (\mathsf{Sim}_\Pi, \mathsf{Ext}_\Pi))$

1. $(\mathsf{crs}, \mathsf{td}, \pi_1) \leftarrow \mathsf{Sim}_\Pi(1^\lambda, S)$.
2. $(\pi_2, (x_1, r_1)) \leftarrow \mathcal{A}(\mathsf{crs}, \pi_1)$.
3. $\mathsf{st} \leftarrow \mathsf{Ext}_\Pi(S, \mathsf{td}, \pi_2)$.
4. If $\pi_2 = \Pi_2(\mathsf{crs}, \pi_1, x_1; r_1)$ then:
   (a) Output $(\mathsf{crs}, \pi_1, f(x_0, x_1))$.
5. If $\pi_2 \neq \Pi_2(\mathsf{crs}, \pi_1, x_1; r_1)$ then:
   (a) Output $(\mathsf{crs}, \pi_1, \mathsf{Sim}_\Pi(\mathsf{st}, \pi_2, x_0))$.

Figure 4: Descriptions of Real_S and Ideal_S experiments.

Proposition 4.3. *Assume black-box access to a two-round oblivious transfer protocol secure against malicious adversaries in the common random/reference string model. There exists a two-party inner protocol for computing any two-party functionality f satisfying Definition 4.1. The computational and communication complexity of the protocol is $\mathsf{poly}(\lambda, |f|)$ where $|f|$ denotes the circuit-size of f.*

4.2 Construction from One-Sided NISC

We note that any one-sided NISC protocol gives rise to a two-party protocol satisfying Definition 4.1. This is because security against malicious receivers is implied by the security of one-sided NISC against malicious receivers. Robust security against semi-malicious senders is implied by security of one-sided NISC against malicious senders. Thus, we get the following corollary.

Corollary 4.4. *Let f be an arbitrary two-party functionality. Assume black-box access to an one-sided NISC protocol that securely computes f. Then, there exists a two-party inner protocol for computing f satisfying Definition 4.1. The computational and communication complexity of the protocol are the same as that of the NISC protocol.*

5 Two-Sided Black-Box NISC

In this section, we give our construction of black-box two-sided NISC protocol. We prove the following theorems.

Theorem 5.1 (Black-box two-sided NISC)**.** *Assume black-box access to a two-round oblivious transfer protocol secure against malicious adversaries in the common random/reference string model. Then, there exists a two-round protocol for securely computing any two-party functionality f against malicious adversaries in the common random/reference string model where both parties get the output of f at the end of the protocol. The computational and communication complexity of the protocol is $\mathsf{poly}(\lambda, |f|)$ where $|f|$ denotes the circuit-size of f.*

Theorem 5.2 (Black-box arithmetic two-sided NISC)**.** *Let $\mathbb{F}$ be a finite field and let f be a two-party functionality that is computable by an arithmetic branching program over $\mathbb{F}$. Assume black-box access to a two-round oblivious linear evaluation (OLE) protocol over $\mathbb{F}$ and an oblivious transfer*

protocol that is secure against malicious adversaries in the common random/reference string model. Then, there exists a two-round protocol for securely computing f against malicious adversaries in the common random/reference string model where both parties get the output of f at the end of the protocol. The computational and communication complexity of the protocol is $\mathsf{poly}(\lambda, |f|)$ where $|f|$ denotes the size of the branching program computing f and the protocol makes black-box use of $\mathbb{F}$.

5.1 Building Blocks

The construction makes use of the following building blocks:

1. A two-round, two client, m server outer MPC protocol $\Psi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ for computing the function f that satisfies security with abort against t server corruptions. We set $t = 2\lambda$ and $m = 3t + 1$. Based on [IKP10, Pas12], we give a construction of such a protocol making black-box use of a PRG in Appendix B where Eval does not involve cryptographic operations.

2. A two-round, two-party inner protocol (see Definition 4.1) $(\mathsf{Setup}_{\Pi_j}, \Pi_{j,1}, \Pi_{j,2}, \mathsf{out}_{\Pi_j})$ that delivers output to the receiver and computes $\mathsf{Eval}(j, \cdot)$ for each $j \in [m]$. From Proposition 4.3 and Corollary 4.4, such a protocol can be constructed making black-box use of a two-round malicious secure OT protocol or an one-sided NISC protocol.

3. A two-round malicious-secure two-party computation protocol $(\mathsf{CRSGen}, \Phi_1, \Phi_2, \mathsf{out}_\Phi)$ for computing the $\mathsf{Sel}_{\lambda,m}$ functionality. The $\mathsf{Sel}_{\lambda,m}$ functionality takes in a string ρ_1 from the receiver, $(\rho_2, (s_1, \ldots, s_m))$ from the sender. It computes $\rho_1 \oplus \rho_2$ and uses it as random tape to select a random multiset (with replacement) K of $[m]$ of size λ. It then outputs $(K, \{s_i\}_{i \in K})$ to the receiver. [IKO+11] gave a two-round black-box protocol for computing $\mathsf{Sel}_{\lambda,m}$ based on two-round malicious-secure OT protocol.

The key lemma that we will prove in this section is the following.

Lemma 5.3. *Assume black-box access to a PRG and the protocols $\{\Pi_j\}_{j \in [m]}$ and Φ as described above. Then, there exists a two-round protocol for securely computing any two-party functionality f against malicious adversaries where both parties get the output of f at the end of the protocol.*

Theorem 5.1 is obtained by instantiating $\{\Pi_j\}_{j \in [m]}$ from Proposition 4.3. To obtain Theorem 5.2, we observe that in the protocols of [IKP10, Pas12], if f is computable by an arithmetic branching program then $\mathsf{Eval}(j, \cdot)$ is computable by a log-depth arithmetic circuit and does not involve any cryptographic operations. Thus, we can instantiate Π_j for each $j \in [m]$ using the one-sided NISC protocol for computing log-depth arithmetic circuits based on two-round malicious secure OLE [IKO+11, CDI+19, DIO21] using Corollary 4.4.

We give the construction of the protocol in Section 5.2 and the proof of security in Section 5.3

5.2 Construction

Let P_0 and P_1 be the two parties with private inputs x_0 and x_1 respectively. The parties additionally have as a common input the description of the function f. We give the formal description of the construction in Figure 5.

- **CRS Generation:** To generate the CRS,
  1. Sample $\mathsf{crs}_j^0, \mathsf{crs}_j^1 \leftarrow \mathsf{Setup}_{\Pi_j}(1^\lambda)$ for each $j \in [m]$.
  2. Sample $\mathsf{crs}^0, \mathsf{crs}^1 \leftarrow \mathsf{CRSGen}(1^\lambda)$.
  3. Output $(\{\mathsf{crs}_j^0, \mathsf{crs}_j^1\}_{j\in[m]}, \mathsf{crs}^0, \mathsf{crs}^1)$.
- **Round-1:** In the first round, each party P_i for $i \in \{0,1\}$ does the following:
  1. It computes $(x_1^i, \ldots, x_m^i) \leftarrow \mathsf{Share}(1^\lambda, i, x_i; r_i)$ for uniformly chosen $r_i \leftarrow \{0,1\}^\lambda$.
  2. For each $j \in [m]$, it samples a uniform random string r_j^i and computes $\pi_{j,1}^i \leftarrow \Pi_{j,1}(\mathsf{crs}_j^i, i, x_j^i; r_j^i)$.
  3. It samples a uniform random string $\rho_1^i \leftarrow \{0,1\}^*$ and computes $\phi_1^i \leftarrow \Phi_1(\mathsf{crs}^i, i, \rho_1^i)$.
  4. It sends $\{\pi_{j,1}^i\}_{j\in[m]}$ and ϕ_1^i to the other party.
- **Round-2:** In the second round, each party P_i for $i \in \{0,1\}$ does the following:
  1. For each $j \in [m]$, it samples a uniform random string t_j^{1-i} and computes $\pi_{j,2}^{1-i} \leftarrow \Pi_{j,2}(\mathsf{crs}_j^{1-i}, i, \pi_{j,1}^{1-i}, x_j^i; t_j^{1-i})$.
  2. For each $j \in [m]$, it sets $s_j^i = (x_j^i, r_j^i, t_j^{1-i})$.
  3. It samples a uniform random string $\rho_2^{1-i} \leftarrow \{0,1\}^*$ and computes $\phi_2^{1-i} \leftarrow \Phi_2(\mathsf{crs}^{1-i}, i, \phi_1^{1-i}, (\rho_2^{1-i}, (s_1^i, \ldots, \mathsf{s}_m^i)))$.
  4. It sends $\{\pi_{j,2}^{1-i}\}_{j\in[m]}$ and ϕ_2^{1-i} to the other party.
- **Output Computation:** To compute the output P_i for $i \in \{0,1\}$ does the following:
  1. It computes $(K^i, \{s_j^{1-i}\}_{j\in K^i})$ using out_Φ on crs^i, ϕ_2^i and the random tape used to generate ϕ_1^i.
  2. For each $j \in K^i$, it:
     (a) Parses s_j^{1-i} as $(x_j^{1-i}, r_j^{1-i}, t_j^i)$.
     (b) Checks if (x_j^{1-i}, r_j^{1-i}) is a consistent input, randomness pair that explains the message $\pi_{j,1}^{1-i}$ and if (x_j^{1-i}, t_j^i) is a consistent input, randomness pair that explains the message $\pi_{j,2}^i$.
  3. If any of the above checks fail, then P_i aborts and outputs $\perp$.
  4. Else, for each $j \in [m]$, it computes $z_j^i := \mathsf{out}_{\Pi_j}(\mathsf{crs}_j^i, \pi_{j,2}^i, (x_j^i, r_j^i))$.
  5. It outputs $\mathsf{Dec}(z_1^i, \ldots, z_m^i, r_i)$.

Figure 5: Black-Box Two-Sided NISC Protocol

5.3 Proof of Security

We give the description of the simulator below and show that the real and the ideal executions are computationally indistinguishable. Since the protocol is symmetric w.r.t. both P_0 and P_1, we assume without loss of generality that P_1 is corrupted by $\mathcal{A}$.

Description of Sim.

- $\mathsf{CRSGen}(1^\lambda)$**:** Sim does the following:
  1. It chooses $(\mathsf{crs}^0, \mathsf{td}^0, \phi_1^0) \leftarrow \mathsf{Sim}_\Phi(1^\lambda, S)$ and $(\mathsf{crs}^1, \mathsf{td}^1) \leftarrow \mathsf{Sim}_\Phi(1^\lambda, R)$.
  2. It samples a uniform multiset K^1 of $[m]$ of size λ.

3. For each $j \in K^1$, it samples $\mathsf{crs}_j^0, \mathsf{crs}_j^1 \leftarrow \mathsf{Setup}_{\Pi_j}(1^\lambda)$.
4. For each $j \notin K^1$, it samples $(\mathsf{crs}_j^0, \mathsf{td}_j^0, \pi_{j,1}^0) \leftarrow \mathsf{Sim}_{\Pi_j}(1^\lambda, S)$ and $(\mathsf{crs}_j^1, \mathsf{td}_j^1) \leftarrow \mathsf{Sim}_{\Pi_j}(1^\lambda, R)$.
5. It outputs $(\{\mathsf{crs}_j^0, \mathsf{crs}_j^1\}_{j\in[m]}, \mathsf{crs}^0, \mathsf{crs}^1)$ as the CRS of the overall protocol.

- **Round-1:** To generate the first round message, Sim does the following:

1. It runs the simulator Sim_Ψ for the outer protocol by corrupting the client P_1 and the set of servers given by K^1. Sim_Ψ provides with $\{x_j^0\}_{j\in K^1}$.
2. For each $j \in K^1$, it computes $\pi_{j,1} \leftarrow \Pi_{j,1}(\mathsf{crs}_j^0, x_j^0; r_j^0)$ for uniformly chosen r_j^0.
3. It sends ϕ_1^0 and $\{\pi_{j,1}^0\}_{j\in[m]}$ to $\mathcal{A}$.
4. It receives the first round message from $\mathcal{A}$. For each $j \notin K^1$, it computes $x_j^1 \leftarrow \mathsf{Ext}_{\pi_j}(\mathsf{td}_j^1, \pi_{j,1}^1)$. It computes $\rho_1^1 \leftarrow \mathsf{Ext}_\Phi(R, \phi_1^1, \mathsf{td}^1)$.

- **Round-2:** To generate the second round message, Sim does the following:

1. It sends $\{x_j^1\}_{j\notin K^1}$ to Sim_Ψ as the first round message from the corrupted client to the honest servers. Sim_Ψ queries the ideal functionality on input x_1 and Sim forwards this query to its own ideal functionality. It forwards the response from the ideal functionality back to Sim_Ψ. Sim_Ψ sends $\{z_j^1\}_{j\notin K^1}$ as the second round message from the honest servers to the corrupted client.
2. For each $j \notin K^1$, it generates $\pi_{j,2}^1 \leftarrow \mathsf{Sim}_{\Pi_j}(R, \mathsf{crs}_j^1, z_j^1, x_j^1)$. For each $j \in K^1$, it generates $\pi_{j,2}^1$ as $\Pi_{j,1}(\mathsf{crs}_j^1, \pi_{j,1}^1, x_j^0; t_j^1)$ for uniformly chosen t_j^1.
3. It generates $\phi_2^1 \leftarrow \mathsf{Sim}_\Phi(R, \{K^1, \{x_j^0, r_j^0, t_j^1\}_{j\in K^1}\})$.
4. It sends ϕ_2^1 and $\{\pi_{j,2}^1\}_{j\in[m]}$ to $\mathcal{A}$.
5. It receives the second round message from $\mathcal{A}$. For each $j \notin K^1$, it computes $\mathsf{st}_j \leftarrow \mathsf{Ext}_{\Pi_j}(S, \mathsf{td}_j^0, \pi_{j,2}^0)$.
6. It also computes $(\rho_2^0, s_1^1, \ldots, s_m^1) \leftarrow \mathsf{Ext}_\Phi(S, \phi_2^0, \mathsf{td}^0)$.

- **Output Computation:** To compute the output, Sim does the following:

1. It chooses a uniform multiset K^0 of $[m]$ size λ and uses it to perform the same checks as done by honest P_0 using $\{s_j^1\}_{j\in K^0}$. If any of the checks fail, it instructs the ideal functionality to deliver $\perp$ to P_0.
2. Otherwise, it initializes an empty set C_1.
3. For each $j \notin K^1$,
   (a) It parses s_j^1 as $(\overline{x}_j^1, r_j^1, t_j^0)$.
   (b) If either $(\overline{x}_j^1, r_j^1)$ is not a consistent input, randomness pair that explains the message $\pi_{j,1}^1$ or if $(\overline{x}_j^1, t_j^0)$ is not a consistent input, randomness pair that explains the message $\pi_{j,2}^1$, then we add j to C_1.
4. If $|C_1| \geq \lambda$, then it instructs the ideal functionality to output $\perp$ to P_1. Otherwise, it instructs Sim_Ψ to adaptively corrupt the set of servers indexed by C_1 and obtains $\{x_j^0\}_{j\in C_1}$.

5. For each $j \in C_1$, it computes z_j^0 as $\mathsf{Sim}_{\Pi_j}(S, \mathsf{st}_j, \pi_{j,2}^0, x_j^0)$. For each $j \in K^1$, it computes z_j^0 as $\mathsf{out}_{\Pi_j}(\mathsf{crs}_j^0, \pi_{j,2}^0, (x_j^0, r_j^0))$.
6. It sends $\{z_j^0\}_{j \in C_1 \cup K^1}$ to Sim_Ψ as the second round message from the corrupted servers to the honest client. If Sim_Ψ instructs the P_0 to abort, then Sim instructs the ideal functionality to deliver $\perp$ to P_0. Otherwise, it instructs it to deliver the output of f to P_0.

Proof of Indistinguishability.

- $\underline{\mathsf{Hyb}_1}$: This corresponds to the output of the real experiment which comprises of the view of $\mathcal{A}$ corrupting P_1 and the output of honest P_0.
- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we make the following changes:
  1. Sample $(\mathsf{crs}^0, \mathsf{td}^0, \phi_1^0) \leftarrow \mathsf{Sim}_\Phi(1^\lambda, S)$.
  2. Obtain ϕ_2^0 from $\mathcal{A}$.
  3. Compute $(\rho_2^0, (s_1^1, \ldots, s_m^1)) \leftarrow \mathsf{Ext}_\Phi(S, \phi_2^0, \mathsf{td}^0)$.
  4. Sample ρ_1^0 uniformly from $\{0,1\}^*$ and sample a multiset K^0 of size λ from $[m]$ using $\rho_1^0 \oplus \rho_2^0$ as the random tape.
  5. Use $(K^0, \{s_j^1\}_{j \in K^0})$ to perform the same checks described in output computation.

  In Lemma 5.4, we show from the simulation security of Φ against corrupted senders that $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$.
- $\underline{\mathsf{Hyb}_3}$: In this hybrid, we make the following changes:
  1. Sample $(\mathsf{crs}^1, \mathsf{td}^1) \leftarrow \mathsf{Sim}_\Phi(1^\lambda, R)$.
  2. Obtain ϕ_1^1 from $\mathcal{A}$.
  3. Compute $\rho_1^1 \leftarrow \mathsf{Ext}_\Phi(R, \phi_1^1, \mathsf{td}^1)$.
  4. Sample a multiset K^1 of size λ from $[m]$ using a random tape ρ^1.
  5. Generate $\phi_2^1 \leftarrow \mathsf{Sim}_\Phi(R, \{K^1, \{s_j^0\}_{j \in K^1}\})$.
  6. Use ϕ_2^1 to generate the final round message in the protocol.

  In Lemma 5.5, we use the simulation security of Φ against corrupted receivers to show that $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$.
- $\underline{\mathsf{Hyb}_4}$: In this hybrid, we make the following changes:
  1. For each $j \in [m]$, we parse s_j^1 as $(\overline{x}_j^1, r_j^1, t_j^0)$.
  2. We initialize an empty set C_1.
  3. For each $j \notin K^1$,
     (a) If either $(\overline{x}_j^1, r_j^1)$ is not a consistent input, randomness pair that explains the message $\pi_{j,1}^1$, or if $(\overline{x}_j^1, t_j^0)$ is not a consistent input, randomness pair that explains the message $\pi_{j,2}^1$, then we add j to C_1.

4. If $|C_1| \geq \lambda$, then we abort and use $\perp$ as the output of honest P_0.

In Lemma 5.6, we show that $\mathsf{Hyb}_3 \approx_s \mathsf{Hyb}_4$.

- $\underline{\mathsf{Hyb}_5}$: In this hybrid, we make the following changes:

  1. Before generating the CRS, we sample a uniform multiset K^1 of $[m]$ with size λ.
  2. We sample $(\mathsf{crs}_j^0, \mathsf{td}_j^0, \pi_{j,1}^0) \leftarrow \mathsf{Sim}_{\Pi_j}(1^\lambda, S)$ for each $j \notin K^1$. We use $\{\mathsf{crs}_j^0\}_{j \notin K^1}$ as part of the CRS and use $\{\pi_{j,1}^0\}_{j \notin K^1}$ to generate the first round message from P_0.
  3. We receive the second round message from $\mathcal{A}$ (that includes $\pi_{j,2}^0$ for each $j \in [m]$) and extract $\{(\overline{x}_j^1, r_j^1, t_j^0)\}_{j \in [m]}$ as before.
  4. For each $j \notin K^1$, we compute $\mathsf{st}_j \leftarrow \mathsf{Ext}_{\Pi_j}(S, \mathsf{td}_j^0, \pi_{j,2}^0)$.
  5. We compute the set C_1 as before.
  6. For each $j \in C_1$, we set $z_j^0 = \mathsf{Sim}_{\Pi_j}(S, \mathsf{st}_j, \pi_{j,2}^0, x_j^0)$.
  7. For each $j \in K^1$, we compute z_j^0 as before.
  8. For each $j \notin C_1 \cup K^1$, we set $z_j^0 = \mathsf{Eval}(j, x_j^0, \overline{x}_j^1)$.

  In Lemma 5.7, we rely on the robust security of Π_j against semi-malicious senders to show that $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$.

- $\underline{\mathsf{Hyb}_6}$: In this hybrid, we make the following changes:

  1. We generate $(\mathsf{crs}_j^1, \mathsf{td}_j^1) \leftarrow \mathsf{Sim}_{\Pi_j}(1^\lambda, R)$ for each $j \notin K^1$.
  2. On receiving $\{\pi_{j,1}^1\}_{j \in [m]}$ from $\mathcal{A}$, we run $\mathsf{Ext}_{\Pi_j}(\mathsf{td}_j^1, \pi_{j,1}^1)$ to obtain x_j^1 for each $j \notin K^1$.
  3. For each $j \notin K^1$, we generate $\pi_{j,2}^1 \leftarrow \mathsf{Sim}_{\Pi_j}(R, \mathsf{crs}_j^1, z_j^1 = \mathsf{Eval}(j, x_j^0, x_j^1), x_j^1)$. We use this to generate the second round message from P_0.

  In Lemma 5.8, we use the security of Π_j against malicious senders for each $j \in [m]$ to show that $\mathsf{Hyb}_5 \approx_c \mathsf{Hyb}_6$.

- $\underline{\mathsf{Hyb}_7}$: In this hybrid, for each $j \notin K^1 \cup C_1$, we use z_j^1 instead of z_j^0 to compute the output of honest P_0. It follows from the correctness of extraction property of $\{\Pi_j\}_{j \notin K^1 \cup C_1}$ that $z_j^0 = z_j^1$ for each $j \notin K^1 \cup C_1$ except with negligible probability and hence, $\mathsf{Hyb}_6 \approx_s \mathsf{Hyb}_7$.

- $\underline{\mathsf{Hyb}_8}$: In this hybrid, we make the following changes:

  1. We start running the simulator Sim_Ψ by corrupting the client P_1 and the set of servers indexed by K^1. We receive $\{x_j^1\}_{j \in K^1}$ from the simulator and use this to generate the first round message from P_0.
  2. On receiving $\{\pi_{j,1}^1\}_{j \in [m]}$ from $\mathcal{A}$, we run $\mathsf{Ext}_{\Pi_j}(\mathsf{td}_j^1, \pi_{j,1}^1)$ to obtain x_j^1 for each $j \notin K^1$. We send $\{x_j^1\}_{j \notin K^1}$ to Sim_Ψ as the first round message from the adversarial client P_1 to the honest servers.
  3. Sim_Ψ queries its ideal functionality on an input x_1 and we forward this to our ideal functionality and respond with $f(x_0, x_1)$.

4. Sim_Ψ provides $\{z_j^1\}_{j\notin K^1}$. We use this to generate $\pi_{j,2}^1 \leftarrow \mathsf{Sim}_{\Pi_j}(z_j^1, x_j^1)$ for each $j \notin K^1$.
5. We receive the second round message from $\mathcal{A}$ and use this to extract $\{s_j^1\}_{j\in[m]}$ as before. We compute the set C_1 and abort if $|C_1| \geq \lambda$.
6. For each $j \notin K^1$, we compute $\mathsf{st}_j \leftarrow \mathsf{Ext}_{\Pi_j}(S, \mathsf{td}_j^0, \pi_{j,2}^0)$.
7. We now instruct Sim_Ψ to adaptively corrupt the set of servers corresponding to C_1 and obtain $\{x_j^0\}_{j\in C_1}$. We then compute $z_j^0 = \mathsf{Sim}_{\Pi_j}(\mathsf{st}_j, \pi_{j,2}^0, x_j^0)$ for each $j \in C_1$. We compute z_j^0 for each $j \in K^1$ as before.
8. We send $\{z_j^0\}_{j\in C_1\cup K^1}$ as the second round message from the corrupted servers to the honest client to Sim_Ψ. If Sim_Ψ instructs the client to abort, we instruct P_0 to do the same. Otherwise, we instruct P_0 to output $f(x_0, x_1)$.

In Lemma 5.9, we use the security of the outer protocol to argue that $\mathsf{Hyb}_7 \approx_c \mathsf{Hyb}_8$. Notice that Hyb_8 is identically distributed to the ideal world using Sim.

Lemma 5.4. *Assuming the simulation security of the protocol Φ against corrupted senders, we have $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$.*

Proof. Assume for the sake of contradiction that Hyb_1 and Hyb_2 are computationally distinguishable with non-negligible advantage. We show that this contradicts the simulation security of the protocol Φ against corrupted senders.

We start interacting with the external challenger and provide a uniformly chosen random string ρ_1^0 as the challenge receiver input. The challenger responds with crs^0. We use this to generate the CRS in the overall protocol. The challenger also sends ϕ_1^0. We use this to generate the first round message in the protocol by sampling the other components of the first round message as in Hyb_1. We generate the second round message as before and obtain the second round message from $\mathcal{A}$. We forward ϕ_2^0 from the second round message received from $\mathcal{A}$ to the external challenger. The external challenger provides with $K^0, \{s_j^1\}_{j\in K^0}$ as the output of the honest P_0. We use this to perform the same checks as described in the output computation. We finally output the view of $\mathcal{A}$ and the output of P_0.

If the messages in the protocol Φ and the CRS and the output of honest P_0 are generated as in the real experiment, then the output of the above reduction is identically distributed to Hyb_1. Else, it is identically distributed to Hyb_2. Thus, if Hyb_2 and Hyb_1 are computationally distinguishable with non-negligible advantage then this breaks the simulation security of Φ against corrupted senders and this is a contradiction. □

Lemma 5.5. *Assuming the simulation security of Sim_Φ against corrupted senders, we have $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$.*

Proof. Assume for the sake of contradiction that Hyb_2 and Hyb_3 are computationally distinguishable with non-negligible advantage. We show that this contradicts the simulation security of Φ against corrupted receivers.

We interact with the external challenger and provide a uniformly chosen ρ_2^1 and $(s_1^0, \ldots, s_m^0)$ as the challenge sender input. The external challenger provides with crs^1 and we use this to generate the CRS of the overall protocol. We start interacting with the adversary and obtain the first round message ϕ_1^1 from it. We forward this to the external challenger. The external challenger provides with the second round message ϕ_2^1 and we use this to generate the second round message in the

overall protocol. We compute the output of honest P_0 as before and finally output the view of $\mathcal{A}$ and the output of the honest P_0.

Note that if the messages in the protocol Φ and the CRS are generated by the external challenger as in the real experiment then the output of the above reduction is distributed identically to Hyb_2. Else, it is distributed identically to Hyb_3. Thus, if Hyb_3 and Hyb_2 are computationally distinguishable with non-negligible advantage then this breaks the simulation security of Φ against corrupted receivers and this is a contradiction. □

Lemma 5.6. $\mathsf{Hyb}_3 \approx_s \mathsf{Hyb}_4$.

Proof. Note that the only difference between Hyb_3 and Hyb_4 is that in Hyb_4 we abort if $|C_1| \geq \lambda$ To show that Hyb_3 and Hyb_4 are statistically close, we prove that if the above condition holds, then in Hyb_3, the checks done by the honest P_0 fails with overwhelming probability.

Note that K^0 is distributed as a random multiset of $[m]$ of size λ. If $C_1 \cap K^0 \neq \emptyset$, then the the honest P_0 in Hyb_3 also aborts. We show that this event happens with overwhelming probability.

$$\begin{aligned}
\Pr[|K^0 \cap C_1| = 0] &= (1 - \frac{|C_1|}{m})^{\lambda} \\
&\leq e^{-|C_1|\lambda/m} \\
&\leq e^{-\lambda^2/m} \\
&\leq e^{-O(\lambda)}
\end{aligned}$$

where the last inequality follows since $m = O(\lambda)$. This completes the proof of the lemma.

□

Lemma 5.7. *Assuming the robust security of Π_j against semi-malicious senders for each $j \in [m]$, we have $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$.*

Proof. Assume for the sake of contradiction that Hyb_4 and Hyb_5 are distinguishable with non-negligible advantage. We sample a uniform multiset K^1 of $[m]$ of size λ. We now show that if Hyb_4 and Hyb_5 are computationally distinguishable then this contradicts the robust security of Π_j against semi-malicious senders for some $j \notin K^1$.

Let $\prec$ be a total order on the set $[m] \setminus K^1$. If Hyb_4 and Hyb_5 are distinguishable with non-negligible advantage, then by a standard averaging argument there exists $\mathsf{Hyb}_{4,j}$ and $\mathsf{Hyb}'_{4,j}$ (described below) that are distinguishable with non-negligible advantage. In both the hybrids, for each $j^* \prec j$, $(\mathsf{crs}^0_{j^*}, \pi^0_{j^*,1})$ is generated as in Hyb_5 whereas for each $j \prec j^*$, $(\mathsf{crs}^0_{j^*}, \pi^0_{j^*,1})$ is generated as in Hyb_4. The only difference is that in $\mathsf{Hyb}_{4,j}$, $(\mathsf{crs}^0_j, \pi^0_{j,1})$ is generated as in Hyb_5 whereas it is generated as in Hyb_4 in $\mathsf{Hyb}'_{4,j}$. We use this to construct an attacker that breaks the robust security of Π_j against semi-malicious senders.

We interact with the external challenger and provide x^0_j as the challenge receiver message. The challenger provides $(\mathsf{crs}^0_j, \pi^0_{j,1})$. We use this to generate the CRS and the first round message of the overall protocol. We receive the second round message from the adversary and use it to extract $\{(\overline{x}^1_j, r^1_j, t^0_j)\}_{j\in[m]}$. We compute the set C_1 as before and abort if $|C_1| \geq \lambda$. For each $j \in C_1$, we send $\overline{x}^1_j$ and an arbitrary t^0_j (that does not explain the messages correctly) along with $\pi^0_{j,2}$ to the external challenger. If $j \notin C_1 \cup K^1$, we send $(\overline{x}^1_j, t^0_j)$ along with $\pi^0_{j,2}$ to the external challenger. We receive the output z^0_j and use this to compute the output of the overall protocol as before.

We note that if $(\mathsf{crs}_j^0, \pi_{j,1}^0, z_j^0)$ was generated by the external challenger as in the Real_S experiment then the output of the above reduction is identically distributed to $\mathsf{Hyb}'_{4,j}$. Else, it is distributed identically to $\mathsf{Hyb}_{4,j}$. Thus, if $\mathsf{Hyb}_{4,j}$ and $\mathsf{Hyb}'_{4,j}$ are distinguishable with non-negligible advantage, then the above reduction breaks the robust security of Π_j against semi-malicious senders with non-negligible advantage and this is a contradiction. □

Lemma 5.8. *Assuming the security of Π_j against malicious receivers for each $j \in [m]$, we have $\mathsf{Hyb}_6 \approx_c \mathsf{Hyb}_5$.*

Proof. Assume for the sake of contradiction that Hyb_5 and Hyb_6 are distinguishable with non-negligible advantage. We sample a uniform multiset K^1 of $[m]$ of size λ. We now show that this contradicts the security of Π_j against malicious receiver for some $j \notin K^1$.

Let $\prec$ be a total order on the set $[m] \setminus K^1$. If Hyb_5 and Hyb_6 are distinguishable with non-negligible advantage then by a standard averaging argument, there exists $\mathsf{Hyb}_{5,j}$ and $\mathsf{Hyb}'_{5,j}$ (described below) that are distinguishable with non-negligible advantage. In both the hybrids, for each $j^* \prec j$, $(\pi_{j^*,2}^1, \mathsf{crs}_{j^*}^1)$ is generated as in Hyb_6 whereas for each $j \prec j^*$, $(\pi_{j^*,2}^1, \mathsf{crs}_{j^*}^1)$ is generated as in Hyb_5. The only difference is that in $\mathsf{Hyb}_{5,j}$, $(\mathsf{crs}_j^1, \pi_{j,2}^1)$ is generated as in Hyb_6 whereas it is generated as in Hyb_5 in $\mathsf{Hyb}'_{5,j}$. We use this to construct an attacker that breaks the security of Π_j against malicious receivers.

We interact with the external challenger and provide x_j^0 as the challenge sender input. We obtain crs_j^1 from the external challenger. We receive $\pi_{j,1}^1$ from the adversary and forward this to the challenger. The challenger responds with $\pi_{j,2}^1$ and we use these to generate the view of the adversary $\mathcal{A}$ and compute the output of P_0 as in $\mathsf{Hyb}'_{5,j}$.

We note that if $(\pi_{j,2}^1, \mathsf{crs}_j^1)$ was generated by the external challenger as in Real_R then the output of the above reduction is identically distributed to $\mathsf{Hyb}'_{5,j}$. Else, it is distributed identically to $\mathsf{Hyb}_{5,j}$. Thus, if $\mathsf{Hyb}_{5,j}$ and $\mathsf{Hyb}'_{5,j}$ are distinguishable with non-negligible advantage, then the above reduction breaks the security of Π_j against malicious receivers with non-negligible advantage and this is a contradiction. □

Lemma 5.9. *Assuming the security of the outer MPC protocol $\Psi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$, we have that $\mathsf{Hyb}_7 \approx_c \mathsf{Hyb}_8$.*

Proof. Assume for the sake of contradiction that Hyb_7 and Hyb_8 are computationally distinguishable with non-negligible advantage. We show that this contradicts the security of outer protocol Ψ.

We start interacting with the outer protocol challenger and provide x_0 as the honest client input. We instruct the challenger to corrupt P_1 and the set of servers indexed by K^1. The challenger provides $\{x_j^0\}_{j \in K^1}$ as the first round message from the honest client to the corrupted servers and we use this to generate the first round message in the protocol. On receiving the first round message from $\mathcal{A}$, we obtain $x_j^1 \leftarrow \mathsf{Ext}_{\Pi_j}(\mathsf{td}_j^1, \pi_{j,1}^1)$ for each $j \notin K^1$ and send $\{x_j^1\}_{j \notin K^1}$ as the first round message from the adversarial client to the honest servers. The challenger replies with $\{z_j^1\}_{j \notin K^1}$. We use this to generate $\pi_{j,2}^1 \leftarrow \mathsf{Sim}_{\Pi_j}(z_j^1, x_j^1)$ for each $j \notin K^1$ and compute the second round message of the overall protocol. We receive the second round message from the adversary. We use this to extract $\{s_j^1\}_{j \in [m]}$ as before. We compute the set C_1 and abort if $|C_1| \geq \lambda$. Additionally, for each $j \in C_1$, we compute $\mathsf{st}_j \leftarrow \mathsf{Ext}_{\Pi_j}(S, \mathsf{td}_j^0, \pi_{j,2}^0)$. We now instruct the challenger to adaptively corrupt the set of servers corresponding to C_1 and obtain $\{x_j^0\}_{j \in C_1}$. We then compute $z_j^0 = \mathsf{Sim}_{\Pi_j}(\mathsf{st}_j, \pi_{j,2}^0, x_j^0)$ for each $j \in C_1$. We compute z_j^0 for each $j \in K^1$ as before.

We send $\{z_j^0\}_{j \in C_1 \cup K^1}$ as the second round message from the corrupted servers to the honest client to the challenger. If the challenger instructs the client to abort, we instruct P_0 to do the same. Otherwise, we instruct P_0 to output whatever is provided by the challenger as the output. We output the view of $\mathcal{A}$ and the output of P_0.

Note that if the messages received from the challenger are computed as in the real execution of the protocol Ψ, then the output of the above reduction is identically to Hyb_7. Else, it is distributed identically to Hyb_8. Hence, if Hyb_7 and Hyb_8 are distinguishable with non-negligible advantage, then the above reduction breaks the security of the outer protocol Ψ with non-negligible advantage and this is a contradiction. □

6 Multiparty Inner Protocol

In this section, we give the definition of a three-round multiparty protocol that satisfies some special properties (known as multiparty inner protocol) and give a construction based on two-round malicious secure oblivious transfer. In the next section, we will use this multiparty inner protocol as the key ingredient to construct a three-round malicious secure protocol for general functionalities.

6.1 Definition

A three-round n-party protocol for computing a function f is given by a tuple of PPT algorithms $(\mathsf{Setup}, \Pi_1, \Pi_2, \Pi_3, \mathsf{out}_\Pi)$ and has the following syntax. Setup algorithm takes in the security parameter 1^λ (encoded in unary) and outputs the common reference string crs. For each $r \in [3]$, Π_r is the r-th round message function that takes in crs, index i of the party, the transcript seen so far (denoted by $\pi(r-1)$), the i-th party's private input x_i, its random tape r_i and outputs π_r^i. out_Π is the public decoder (see [ABG+20] for the definition of a publicly decodable MPC) that takes in the transcript of the three rounds $\pi(3)$ and outputs $f(x_1, \ldots, x_n)$.

Definition 6.1. *A three-round n-party protocol $(\mathsf{Setup}, \Pi_1, \Pi_2, \Pi_3, \mathsf{out}_\Pi)$ for computing a function f is said to be a multiparty inner protocol with publicly decodable transcript if it satisfies:*

- ***Correctness:*** *For any choice of inputs $x_1, \ldots, x_n$, we have:*

$$\Pr[\mathsf{out}_\Pi(\pi(3)) = f(x_1, \ldots, x_n)] = 1$$

  where $\pi(3)$ is the transcript generated in the first three rounds of the protocol.

- ***Security:*** *For any subset $M \subset [n]$ of the parties, there exists a (stateful) PPT simulator Sim_Π such that for any (stateful) non-uniform PPT adversary $\mathcal{A}$ corrupting the set of parties given by M and for any set $\{x_i\}_{i \in [n] \setminus M}$ of the honest party inputs, we have:*

$$\Big|\Pr[\mathsf{Real}(1^\lambda, M, \mathcal{A}, \{x_i\}_{i \in [n] \setminus M}) = 1] - \Pr[\mathsf{Ideal}(1^\lambda, M, \mathcal{A}, \{x_i\}_{i \in [n] \setminus M}, \mathsf{Sim}_\Pi) = 1]\Big| \leq \mathsf{negl}(n)$$

  where Real *and* Ideal *experiments are described in Figure 6.*

Proposition 6.2. *Assume black-box access to a two-round oblivious transfer protocol that is secure against malicious adversaries in the common random/reference string model. Then, there exists*

a three-round inner protocol for computing any n-party functionality f satisfying Definition 6.1. The computational and communication complexity of this protocol is $\mathsf{poly}(\lambda, n, |f|)$ where $|f|$ is the circuit-size of f.

$\mathsf{Real}(1^\lambda, M, \mathcal{A}, \{x_i\}_{i\in[n]\setminus M})$	$\mathsf{Ideal}(1^\lambda, M, \mathcal{A}, \{x_i\}_{i\in[n]\setminus M}, \mathsf{Sim}_\Pi)$
1. $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$.	1. $(\mathsf{crs}, \mathsf{td}, \{\pi_1^i\}_{i\in[n]\setminus M}) \leftarrow \mathsf{Sim}_\Pi(1^\lambda, M)$.
2. Compute $\pi_1^i \leftarrow \Pi_1(\mathsf{crs}, x_i; r_i)$ for each $i \in [n]\setminus M$ where $r_i \leftarrow \{0,1\}^*$.	
3. $\{\pi_1^i\}_{i\in M} \leftarrow \mathcal{A}(\mathsf{crs}\{\pi_t^i\}_{i\in[n]\setminus M})$.	2. $\{\pi_1^i\}_{i\in M} \leftarrow \mathcal{A}(\mathsf{crs}, \{\pi_1^i\}_{i\in[n]\setminus M})$.
4. Set $\pi(1) = \{\pi_1^i\}_{i\in[n]}$.	3. Set $\pi(1) = \{\pi_1^i\}_{i\in[n]}$.
5. Compute $\pi_2^i \leftarrow \Pi_2(\mathsf{crs}, \pi(1), x_i; r_i)$ for each $i \in [n]\setminus M$.	4. $\{\pi_2^i\}_{i\in[n]\setminus M} \leftarrow \mathsf{Sim}_\Pi(\pi(1), \mathsf{td})$.
6. $\{\pi_2^i, (x_i, r_i)\}_{i\in M} \leftarrow \mathcal{A}(\{\pi_2^i\}_{i\in[n]\setminus M})$.	5. $\{\pi_2^i, (x_i, r_i)\}_{i\in M} \leftarrow \mathcal{A}(\{\pi_2^i\}_{i\in[n]\setminus M})$.
7. Set $\pi(2) = \pi(1) \cup \{\pi_2^i\}_{i\in[n]}$.	6. Set $\pi(2) = \pi(1) \cup \{\pi_2^i\}_{i\in[n]}$.
8. Compute $\pi_3^i \leftarrow \Pi_3(\mathsf{crs}, \pi(2), x_i; r_i)$ for each $i \in [n]\setminus M$.	7. For each $i \in M$, if (x_i, r_i) is a valid input/randomness pair w.r.t. $\pi(2)$: (a) $\{\pi_3^i\}_{i\in[n]\setminus M} \leftarrow \mathsf{Sim}_\Pi(\pi(2), f(x_1, \ldots, x_n))$. 8. Else, (a) $\{\pi_3^i\}_{i\in[n]\setminus M} \leftarrow \mathsf{Sim}_\Pi(\pi(2), \mathsf{td}, \{x_i\}_{i\in[n]\setminus M})$.
9. $\{\pi_3^i\}_{i\in M} \leftarrow \mathcal{A}(\{\pi_3^i\}_{i\in[n]\setminus M})$.	9. $\{\pi_3^i\}_{i\in M} \leftarrow \mathcal{A}(\{\pi_3^i\}_{i\in[n]\setminus M})$.
10. Set $\pi(3) = \pi(2) \cup \{\pi_3^i\}_{i\in[n]}$	10. Set $\pi(3) = \pi(2) \cup \{\pi_3^i\}_{i\in[n]}$.
11. Output $\mathcal{A}(\mathsf{crs}, \pi(3))$.	11. Output $\mathcal{A}(\mathsf{crs}, \pi(3))$.

Figure 6: Descriptions of Real and Ideal experiments.

In Section 6.2, we give a protocol for computing a special multiparty functionality, namely, 3MULTPlus and in Section 6.3, we show how to bootstrap this protocol to arbitrary functionalities.

6.2 Protocol for 3MULTPlus

In this subsection, we give a protocol for computing the 3MULTPlus functionality that makes black-box use of a two-round OT protocol $(\mathsf{Setup}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{out}_{\mathsf{OT}})$. The 3MULTPlus is a special multiparty functionality that takes $(x_i, y_i) \in \{0,1\} \times \{0,1\}$ from party P_i for each $i \in [3]$ and outputs $x_1 \cdot x_2 \cdot x_3 + y_1 + y_2 + y_3$ to all the parties (where $+$ and $\cdot$ refer to the $\mathbb{F}_2$ operations). The construction for computing 3MULTPlus appears in Figure 7. This protocol is same as that in [PS21] except that it uses a two-round malicious secure OT protocol.

6.2.1 Proof of Security

The correctness follows from the same argument as given in [PS21] and we now show that this construction satisfies the security property in Definition 6.1. Let $\mathcal{A}$ be an adversary corrupting a subset M of the parties and let H denote the set of uncorrupted parties.

- **CRS Generation:** For each $i \in [3]$, generate $\mathsf{crs}_i \leftarrow \mathsf{Setup}(1^\lambda)$. Output $\{\mathsf{crs}_i\}_{i\in[3]}$ as the CRS.
- **Round-1:** In the first round,
  1. P_1 computes $\mathsf{otr} := \mathsf{OT}_1(\mathsf{crs}_1, x_1; r)$ where r is uniformly chosen.
  2. P_2 chooses random bits $x_{2,0}, x_{2,1} \leftarrow \{0,1\}$ subject to $x_2 = x_{2,0} + x_{2,1}$. It computes $\mathsf{otr}_0 := \mathsf{OT}_1(\mathsf{crs}_2, x_{2,0}; r_0)$ and $\mathsf{otr}_1 := \mathsf{OT}_1(\mathsf{crs}_2, x_{2,1}; r_1)$ where r_0, r_1 are uniformly chosen.
  3. P_3 computes $\mathsf{otr}_3 := \mathsf{OT}_1(\mathsf{crs}_3, x_3; r_3)$ where r_3 is uniformly chosen.
  4. P_1 broadcasts otr, P_2 broadcasts $(\mathsf{otr}_0, \mathsf{otr}_1)$, and P_3 broadcasts otr_3.
  5. For every $i \in [3]$, P_i chooses a random additive secret sharing of 0 given by $(\delta_1^i, \delta_2^i, \delta_3^i)$ and sends the share δ_j^i to party P_j for $j \in [3] \setminus \{i\}$ via private channels. We note that we can simulate a single round of private channel messages in two-rounds over public channels by making use of a two-round oblivious transfer.
- **Round-2:** In the second round,
  1. P_2 computes $\mathsf{ots} \leftarrow \mathsf{OT}_2(\mathsf{crs}_1, \mathsf{otr}, (x_{2,0}, r_0), (x_{2,1}, r_1))$. It then chooses random bits $x_{2,0,0}, x_{2,0,1} \leftarrow \{0,1\}$ subject to $x_{2,0} = x_{2,0,0} + x_{2,0,1}$. It computes $\mathsf{ots}_3 \leftarrow \mathsf{OT}_2(\mathsf{crs}_3, \mathsf{otr}_3, x_{2,0,0}, x_{2,0,1})$.
  2. P_3 chooses random bits $x_{3,0}, x_{3,1} \leftarrow \{0,1\}$ subject to $x_3 = x_{3,0} + x_{3,1}$. For each $b \in \{0,1\}$, it first computes $\mathsf{ots}_b \leftarrow \mathsf{OT}_2(\mathsf{crs}_2, \mathsf{otr}_b, x_{3,0}, x_{3,1})$. It then computes $\overline{\mathsf{ots}} \leftarrow \mathsf{OT}_2(\mathsf{crs}_1, \mathsf{otr}, \mathsf{ots}_0, \mathsf{ots}_1)$.
  3. P_2 sends ots to P_1 via private channel and ots_3 to P_3 via private channel. P_3 sends $\overline{\mathsf{ots}}$ to P_1 via private channel. As noted before, single round of private channel message can be simulated in two-rounds over a public channel making use of a two-round oblivious transfer.
- **Round-3:** In the last round,
  1. For each $i \in [3]$, P_i computes $\delta_i = \delta_i^1 + \delta_i^2 + \delta_i^3$.
  2. P_2 sets $z_2 := x_{2,0,0} + y_2 + \delta_2$.
  3. P_3 computes $x_{2,0,x_3} := \mathsf{out}_{\mathsf{OT}}(\mathsf{ots}_3, (x_3, r_3))$ and sets $z_3 = x_{2,0,x_3} + x_{3,0} + y_3 + \delta_3$.
  4. P_1 computes $(x_{2,x_1}, r_{x_1}) := \mathsf{out}_{\mathsf{OT}}(\mathsf{ots}, (x_1, r))$ and $\mathsf{ots}_{x_1} := \mathsf{out}_{\mathsf{OT}}(\overline{\mathsf{ots}}, (x_1, r))$. It then computes $x_{3,x_{2,x_1}} := \mathsf{out}_{\mathsf{OT}}(\mathsf{ots}_{x_1}, (x_{2,x_1}, r_{x_1}))$. It then sets $z_1 := x_{3,x_{2,x_1}} + y_1 + \delta_1$.
  5. P_1 broadcasts z_1, P_2 broadcasts z_2 and P_3 broadcasts z_3.
- **Output:** Every party outputs $z_1 + z_2 + z_3$.

Figure 7: Description of the three-round protocol for 3MULTPlus taken verbatim from [PS21]

Description of Simulator. Sim does the following:

- **CRS Generation**: To generate the CRS:
  1. For $i \in \{1,3\}$, if $P_i \in M$, Sim generates $(\mathsf{crs}_i, \mathsf{td}_i) \leftarrow \mathsf{Sim}_{\mathsf{OT}}(1^\lambda, R)$. Else, it generates $(\mathsf{crs}_i, \mathsf{td}_i, \mathsf{otr}) \leftarrow \mathsf{Sim}_{\mathsf{OT}}(1^\lambda, S)$ if $i = 1$ and generates $(\mathsf{crs}_i, \mathsf{td}_i, \mathsf{otr}_3) \leftarrow \mathsf{Sim}_{\mathsf{OT}}(1^\lambda, S)$ if $i = 3$.
  2. If $P_2 \in M$, Sim generates $(\mathsf{crs}_2, \mathsf{td}_2) \leftarrow \mathsf{Sim}_{\mathsf{OT}}(1^\lambda, R)$. Else, it generates $\mathsf{crs}_2 \leftarrow \mathsf{Setup}(1^\lambda)$.
  3. It outputs $(\{\mathsf{crs}_i\}_{i\in[3]})$ as the CRS.
- **Round-1:**
  1. If $P_2 \in H$, it samples a random bit $x \leftarrow \{0,1\}$ and generates $\mathsf{otr}_0 \leftarrow \mathsf{OT}_1(\mathsf{crs}_2, x; r_0)$ and $\mathsf{OT}_2(\mathsf{crs}_2, x; r_1)$ for uniformly chosen r_0, r_1.

2. If P_1 or $P_3 \in H$, Sim uses otr or otr_3 as the first round message.
3. For each corrupted party k, Sim sends a random bit as δ_k^i on behalf of the honest party i.
4. Sim receives the first round message from $\mathcal{A}$.

- **Round-2**:

1. If $P_1 \in M$, Sim sets $x_1 \leftarrow \mathsf{Ext}_{\mathsf{OT}}(R, \mathsf{td}_1, \mathsf{otr})$. Additionally, if $P_2 \in H$, it sets $\mathsf{ots} \leftarrow \mathsf{OT}_2(\mathsf{crs}_1, \mathsf{otr}, (x, r_{x_1}), (x, r_{x_1}))$.
2. If $P_2 \in M$, Sim sets $x_{2,0} \leftarrow \mathsf{Ext}_{\mathsf{OT}}(R, \mathsf{td}_2, \mathsf{otr}_0)$ and $x_{2,1} \leftarrow \mathsf{Ext}_{\mathsf{OT}}(R, \mathsf{td}_2, \mathsf{otr}_1)$. If $P_3 \in H$ and $P_1 \in M$, it computes $\mathsf{ots}_{x_1} \leftarrow \mathsf{OT}_2(\mathsf{crs}_2, \mathsf{otr}_{x_1}, x_{3,x_{2,x_1}}, x_{3,x_{2,x_1}})$ where $x_{3,x_{2,x_1}}$ is uniformly chosen. It also computes $\overline{\mathsf{ots}} \leftarrow \mathsf{OT}_2(\mathsf{crs}_1, \mathsf{otr}, \mathsf{ots}_{x_1}, \mathsf{ots}_{x_1})$.
3. If $P_2 \in H$ and $P_3 \in M$, Sim sets $x_3 \leftarrow \mathsf{Ext}_{\mathsf{OT}}(R, \mathsf{td}_3, \mathsf{otr}_3)$. It samples a random bit $x_{2,0,x_3}$ and computes $\mathsf{ots}_3 \leftarrow \mathsf{OT}_2(\mathsf{crs}_3, \mathsf{otr}_3, (x_{2,0,x_3}, x_{2,0,x_3}))$.
4. It sends the second round message on behalf of the honest parties to $\mathcal{A}$ and it receives the second round message from $\mathcal{A}$.

- **Round-3:**

1. If the the input, randomness provided by $\mathcal{A}$ is consistent with the messages generated by the malicious parties, then Sim computes $\{z_i\}_{i \in M}$ using this information. It samples $\{z_i\}_{i \in H}$ uniformly such that $z_1 + z_2 + z_3 = x_1 x_2 x_3 + y_1 + y_2 + y_3$.
2. Else,
   (a) For each $j \in H$, it sets δ_j^i such that $(\delta_1^i, \delta_2^i, \delta_3^i)$ form a secret sharing of 0. It computes $\delta_i = \delta_i^1 \oplus \delta_i^2 \oplus \delta_i^3$.
   (b) If $P_2 \in H$, it sets $x_{2,x_1} = x$ and computes $x_{2,1-x_1} = x_{2,x_1} \oplus x_2$. It then sets $x_{2,0,1-x_3} = x_{2,0,x_3} \oplus x_{2,0}$. It finally sets $z_2 = x_{2,0,0} + y_2 + \delta_2$.
   (c) If $P_3 \in H$, it computes $(x_{2,0,0}, x_{2,0,1}) \leftarrow \mathsf{Ext}_{\mathsf{OT}}(S, \mathsf{td}_3, \mathsf{ots}_3)$ if $P_2 \in M$. It then computes x_{2,x_1} as described in Step-2 of round-2 if $P_2 \in M$ or as in Step-2.(b) of round-3 if $P_2 \in H$. It sets $x_{3,1-x_{2,x_1}} = x_3 - x_{3,x_{2,x_1}}$ (if $P_1 \in M$, use $x_{3,x_{2,x_1}}$ sampled before, else, use a freshly sampled bit). It then sets $z_3 = x_{2,0,x_3} + x_{3,0} + y_3 + \delta_3$.
   (d) If $P_1 \in H$, it computes $(x_{2,0}, r_0), (x_{2,1}, r_1) \leftarrow \mathsf{Ext}_{\mathsf{OT}}(S, \mathsf{td}_1, \mathsf{ots})$ if $P_2 \in M$ and if $P_3 \in M$, it computes $(\mathsf{ots}_0, \mathsf{ots}_1) \leftarrow \mathsf{Ext}_{\mathsf{OT}}(S, \mathsf{td}_1, \overline{\mathsf{ots}})$. It computes $x_{3,x_{2,x_1}} := \mathsf{out}_{\mathsf{OT}}(\mathsf{ots}_{x_1}, (x_{2,x_1}, r_{x_1}))$. We set $z_1 := x_{3,x_{2,x_1}} + y_1 + \delta_1$.

Proof of Indistinguishability.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the view of the adversary and the outputs of the honest parties (which is deterministically derived from the transcript of the protocol) in the real world execution of the protocol.

- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we make the following changes if $P_1 \in M$:

1. We set $(\mathsf{crs}_1, \mathsf{td}_1) \leftarrow \mathsf{Sim}_{\mathsf{OT}}(1^\lambda, R)$.
2. We receive the first round message from $\mathcal{A}$ and compute $x_1 \leftarrow \mathsf{Ext}_{\mathsf{OT}}(R, \mathsf{td}_1, \mathsf{otr})$.

3. If $P_2 \in H$, we set $\mathsf{ots} \leftarrow \mathsf{OT}_2(\mathsf{crs}_1, \mathsf{otr}, (x_{2,x_1}, r_{x_1}), (x_{2,x_1}, r_{x_1}))$ instead of $\mathsf{OT}_2(\mathsf{otr}, (x_{2,0}, r_0), (x_{2,1}, r_1))$.
4. If $P_3 \in H$, we set $\overline{\mathsf{ots}} \leftarrow \mathsf{OT}_2(\mathsf{crs}_1, \mathsf{ots}_{x_1}, \mathsf{ots}_{x_1})$.

In Lemma 6.3, we use the security of the OT protocol against malicious receivers to show that Hyb_0 and Hyb_1 are computationally indistinguishable.

- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we make the following changes if $P_2 \in M$:

1. We set $(\mathsf{crs}_2, \mathsf{td}_2) \leftarrow \mathsf{Sim}_{\mathsf{OT}}(1^\lambda, R)$.
2. We receive the first round message from $\mathcal{A}$ and compute $x_{2,0} \leftarrow \mathsf{Ext}_{\mathsf{OT}}(R, \mathsf{td}_2, \mathsf{otr}_0)$ and $x_{2,1} \leftarrow \mathsf{Ext}_{\mathsf{OT}}(R, \mathsf{td}_2, \mathsf{otr}_1)$.
3. If $P_3 \in H$, we set $\mathsf{ots}_{x_1} \leftarrow \mathsf{OT}_2(\mathsf{crs}_2, \mathsf{otr}_{x_1}, x_{3,x_{2,x_1}}, x_{3,x_{2,x_1}})$.

Via an identical argument as in Lemma 6.3, we can rely on the security of the OT protocol against malicious receivers to show that Hyb_1 and Hyb_2 are computationally distinguishable.

- $\underline{\mathsf{Hyb}_3}$: In this hybrid, we make the following changes if $P_3 \in M$:

1. We set $(\mathsf{crs}_3, \mathsf{td}_3) \leftarrow \mathsf{Sim}_{\mathsf{OT}}(1^\lambda, R)$.
2. We receive the first round message from $\mathcal{A}$ and compute $x_3 \leftarrow \mathsf{Ext}_{\mathsf{OT}}(R, \mathsf{td}_3, \mathsf{otr}_3)$.
3. If $P_2 \in H$, we set $\mathsf{ots}_3 \leftarrow \mathsf{OT}_2(\mathsf{crs}_3, \mathsf{otr}_3, x_{2,0,x_3}, x_{2,0,x_3})$.

Via an identical argument as in Lemma 6.3, we can rely on the security of the OT protocol against malicious receivers to show that Hyb_2 and Hyb_3 are computationally distinguishable.

- $\underline{\mathsf{Hyb}_4}$: Skip this hybrid change if $P_2 \in M$. In this hybrid, we make the following changes:

1. If $((x_i, y_i), s_i)_{i \in M}$ provided by the adversary $\mathcal{A}$ at the end of round-2 (where s_i is the random tape used by $\mathcal{A}$) is consistent, then we compute z_1 and z_3 using this information and we set $z_2 = x_1 x_2 x_3 + y_1 + y_2 + y_3 - z_1 - z_3$.
2. Else, we obtain (x_2, y_2). We set $x_{2,1-x_1} = x_{2,x_1} \oplus x_2$. We set $x_{2,0,1-x_3} = x_{2,0,x_3} \oplus x_{2,0}$. We set $z_2 = x_{2,0,0} + y_2 + \delta_2$.

This change is only syntactic and hence, $\mathsf{Hyb}_3 \equiv \mathsf{Hyb}_4$.

- $\underline{\mathsf{Hyb}_5}$: Skip this hybrid if $P_2 \in M$. In this hybrid, we make the following changes:

1. We sample a random bit $x \leftarrow \{0,1\}$ and generate $\mathsf{otr}_0 \leftarrow \mathsf{OT}_1(\mathsf{crs}_2, x; r_0)$ and $\mathsf{otr}_1 \leftarrow \mathsf{OT}_1(\mathsf{crs}_2, x; r_1)$.
2. We generate $\mathsf{ots} \leftarrow \mathsf{OT}_2(\mathsf{crs}_1, (x, r_{x_1}), (x, r_{x_1}))$.
3. If $((x_i, y_i), s_i)_{i \in M}$ provided by the adversary $\mathcal{A}$ at the end of round-2 (where s_i is the random tape used by $\mathcal{A}$) is inconsistent, we set $x_{2,x_1} = x$ and do the exact same steps described in Step-2 of Hyb_4.

In Lemma 6.4, we rely on the indistinguishability of the receiver messages of the OT protocol to show that $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$.

- $\underline{\mathsf{Hyb}_6}$: Skip this hybrid if $P_3 \in M$. In this hybrid, we make the following changes:

1. We sample $(\mathsf{crs}_3, \mathsf{td}_3, \mathsf{otr}_3) \leftarrow \mathsf{Sim}_{\mathsf{OT}}(1^\lambda, S)$.
2. If $((x_i, y_i), s_i)_{i \in M}$ provided by the adversary $\mathcal{A}$ at the end of round-2 (where s_i is the random tape used by $\mathcal{A}$) is consistent, then we compute z_1 using this information and if $P_2 \in M$, we also compute z_2 using this information and we set $z_3 = x_1x_2x_3 + y_1 + y_2 + y_3 - z_1 - z_2$. If $P_2 \notin M$, we sample z_3 uniformly at random.
3. Else, we generate $(x_{2,0,0}, x_{2,0,1}) \leftarrow \mathsf{Ext}_{\mathsf{OT}}(S, \mathsf{td}_3, \mathsf{ots}_3)$ if $P_2 \in M$. We compute x_{2,x_1} as in Step 2 of Hyb_2 if $P_2 \in M$ and if $P_2 \in H$, we compute it as in step-3 of Hyb_5. We set $x_{3,1-x_{2,x_1}} = x_3 - x_{3,x_{2,x_1}}$. We then set $z_3 = x_{2,0,x_3} + x_{3,0} + y_3 + \delta_3$.

In Lemma 6.5, we rely on the security of the OT protocol against malicious senders to show that $\mathsf{Hyb}_5 \approx_c \mathsf{Hyb}_6$.

- $\underline{\mathsf{Hyb}_7}$: Skip this hybrid if $P_1 \in M$. In this hybrid, we make the following changes:

1. We sample $(\mathsf{crs}_1, \mathsf{td}_1, \mathsf{otr}) \leftarrow \mathsf{Sim}_{\mathsf{OT}}(1^\lambda, S)$.
2. If $((x_i, y_i), s_i)_{i \in M}$ provided by the adversary $\mathcal{A}$ at the end of round-2 (where s_i is the random tape used by $\mathcal{A}$) is consistent, then we compute z_2 if $P_2 \in M$ and z_3 if $P_3 \in M$ using this information. If either P_2 or P_3 is in H, we sample z_1 uniformly at random. Else, we set $z_1 = x_1x_2x_3 + y_1 + y_2 + y_3 - z_2 - z_3$.
3. Else, if $P_2 \in M$, we compute $(x_{2,0}, r_0), (x_{2,1}, r_1) \leftarrow \mathsf{Ext}_{\mathsf{OT}}(S, \mathsf{td}_1, \mathsf{ots})$ and if $P_3 \in M$, we compute $(\mathsf{ots}_0, \mathsf{ots}_1) \leftarrow \mathsf{Ext}_{\mathsf{OT}}(S, \mathsf{td}_1, \overline{\mathsf{ots}})$. We compute $x_{3,x_{2,x_1}} := \mathsf{out}_{\mathsf{OT}}(\mathsf{ots}_{x_1}, (x_{2,x_1}, r_{x_1}))$. We set $z_1 := x_{3,x_{2,x_1}} + y_1 + \delta_1$.

Via a similar argument to Lemma 6.5, we can rely on the security of the OT protocol against malicious receivers to show that Hyb_6 and Hyb_7 are computationally indistinguishable. We note that Hyb_7 is identical to the ideal world using Sim.

Lemma 6.3. *Assuming the security of the OT protocol against malicious receivers, we have* $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$.

Proof. Assume for the sake of contradiction that Hyb_0 and Hyb_1 are computationally distinguishable with non-negligible advantage. We show that this contradicts the security of the OT protocol against malicious receivers.

We start interacting with the challenger and receive crs_1. We use this to generate the first round message on behalf of the honest parties and receive otr from $\mathcal{A}$. We send this message along with $((x_{2,0}, r_0), (x_{2,1}, r_1))$ as the first set of challenge sender inputs and $(\mathsf{ots}_0, \mathsf{ots}_1)$ as the second set of challenge sender inputs. The challenger provides ots and $\overline{\mathsf{ots}}$ and we use it to generate the second round message in the protocol. We compute the output of honest parties as before and finally output the view of $\mathcal{A}$ and the output of honest parties.

If the view generated by the challenger is distributed identically to the real execution of the OT protocol, then the output of the above reduction is identically distributed to Hyb_0. Else, it is distributed identically to Hyb_1. Since we assumed that Hyb_0 and Hyb_1 are distinguishable with non-negligible advantage, the above reduction breaks the receiver security of the OT protocol with non-negligible advantage and this is a contradiction. □

Lemma 6.4. *Assuming the indistinguishability of the receiver message of the OT protocol, we have* $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$.

Proof. Assume for the sake of contradiction that Hyb_4 and Hyb_5 are computationally distinguishable via a distinguisher D with non-negligible advantage $\epsilon(\lambda)$. We show that this contradicts the indistinguishability of the receiver message of the OT protocol.

We choose a random bit $\alpha \leftarrow \{0,1\}$. We send $x_{2,\alpha}$ and $x_{2,1-\alpha}$ as the receiver challenge bits to the external challenger and the external challenger responds with $\mathsf{crs}_2, \mathsf{otr}_\alpha$. We generate $\mathsf{otr}_{1-\alpha}$ as before and we use this to generate the first round message on behalf of P_2. On receiving otr in the first round, we compute $x_1 = \mathsf{Ext}_{\mathsf{OT}}(R, \mathsf{td}_1, \mathsf{otr})$ and we check if $\alpha = x_1$. If it was the case, we output a random bit to the challenger. Else, if $\alpha \neq x_1$, then we generate $\mathsf{ots} \leftarrow \mathsf{OT}_2(\mathsf{crs}_1, \mathsf{otr}, (x_{2,1-\alpha}, r_{1-\alpha}), (x_{2,1-\alpha}, r_{1-\alpha}))$. We compute the third round message of P_2 as described in Hyb_4 and run the distinguisher D on the view of the adversary and the output of the honest parties. We output whatever the distinguisher D outputs to external challenger.

Let p be the probability that $\alpha \neq x_1$. Note that in Hyb_4, $p = 1/2$. If $p \leq 1/3$ in Hyb_5, then we can use this to directly break the indistinguishability of the receiver messages of the OT protocol. Hence, let us assume that $p \geq 1/3$. Note that if $\alpha \neq x_1$, then the input provided to the distinguisher D by the above reduction is identical to Hyb_4, if the external challenger chose $x_{2,\alpha}$. Else, it is distributed identically to Hyb_5. Thus, probability that the above reduction correctly guesses whether the external challenger chose $x_{2,\alpha}$ or $x_{2,1-\alpha}$ is $(1-p)(1/2) + p(1/2 + \epsilon(\lambda)) = 1/2 + p\epsilon(\lambda) \geq 1/2 + \epsilon(\lambda)/3$ (which is a contradiction). □

Lemma 6.5. *Assuming the security of the OT protocol against malicious senders, we have* $\mathsf{Hyb}_5 \approx_c \mathsf{Hyb}_6$.

Proof. Assume for the sake of contradiction that Hyb_5 and Hyb_6 are computationally distinguishable with non-negligible advantage. We show that this contradicts the security of the OT protocol against malicious senders.

We start interacting with the external challenger and provide x_3 as the challenge receiver choice bit. The challenger provides $\mathsf{crs}_3, \mathsf{otr}_3$. We use this to generate the first round message in the protocol. At the end of the second round, if the inputs provided by $\mathcal{A}$ are consistent, then we compute z_3 as described in Step-2 of Hyb_6. Else, we forward ots_3 to the challenger and obtain $x_{2,0,x_3}$ from it. We then use it to compute z_3 as described in Step-3 of Hyb_6. We finally output the view of $\mathcal{A}$ and the outputs of all the honest parties.

We first observe that if the inputs, random tape provided by the adversary at the end of round-2 is consistent, then if $P_2 \in H$, then z_3 is distributed identically to a uniformly chosen random bit. This is because $\delta_1, \delta_2, \delta_3$ form a random secret sharing of 0. Further, if the view generated by the external challenger is distributed as per the real execution of the OT protocol then the output of the reduction is distributed identically to Hyb_5. Else, it is distributed identically to Hyb_6. Since we assumed that Hyb_5 and Hyb_6 are computationally distinguishable with non-negligible advantage, we conclude that the above reduction breaks the security of the OT protocol against malicious senders and this is a contradiction. □

6.3 Protocol for General Functions

The protocol for general functions follows via parallel executions of the protocol for 3MULTPlus functionality. Specifically, the following theorem given in [BGI+18, GIS18, ABG+20] for the case of semi-honest adversaries directly extends to the security property given in Definition 6.1.

Theorem 6.6 ([BGI+18, GIS18, ABG+20]). *Let f be an n-party functionality. There exists a protocol Π_f for securely computing f satisfying Definition 6.1, where Π_f makes black-box use of the protocol for* 3MULTPlus *functionality satisfying Definition 6.1. The protocol Π_f can either be: (1) computationally secure using a black-box PRG, where the complexity of the parties is polynomial in n, the security parameter λ and the circuit size of f, or alternatively (2) perfectly secure, where the complexity of the parties is polynomial in n and the branching program size of f.*

Note that at the end of round-2, using the information provided by the adversary, we can ascertain if the adversary is using consistent inputs across different parallel executions and if the PRG computations are correct. If both conditions pass, we use the simulator for each parallel execution with the corresponding output. Else, we corrupt all the parallel executions and use the honest party's inputs to simulate the last round message.

7 Round-Optimal Black-Box MPC

In this section, we give a construction of a three-round MPC protocol that makes black-box use of two-round malicious secure oblivious transfer. The round-optimality of this construction follows from [ABG+20]. We prove the following theorem.

Theorem 7.1 (Black-box three-round MPC). *Assume black-box access to a two-round oblivious transfer protocol that is secure against malicious adversaries in the common random/reference string model. Then, there exists a three-round protocol for computing any n-party functionality f in the common random/reference string model that satisfies security with unanimous abort against malicious adversaries corrupting an arbitrary subset of the parties. The protocol works over broadcast channels and its computational and communication complexity is $\mathsf{poly}(\lambda, n, |f|)$ where $|f|$ is the circuit-size of f.*

7.1 Building Blocks

The construction makes use of the following building blocks:

1. A two-round, n client, m server outer MPC protocol $\Psi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ for computing an augmented function g that satisfies privacy with knowledge of outputs property against t server corruptions.[7] The inputs to g are given by $\{(x_i, k_i)\}_{i\in[n]}$ where k_i is a key for a one-time MAC scheme. The output of g is given by $y = f(x_1, \ldots, x_n)$ along with $\{\mathsf{MAC}(k_i, y)\}_{i\in[n]}$. We set $t = 2\lambda n^3$ and $m = 3t + 1$. This protocol has a public decoder Dec that takes in the second round messages from all the servers and computes the output of the parties (without any additional secrets). Such a protocol was constructed in [IKP10, Pas12]. As in Section 5, we require Eval to be information-theoretic and not involve any cryptographic operations. As noted in [IKSS21], we can delegate the PRG computations made by the servers to the client and ensure that the computation done by the servers do not involve any cryptographic operations. In our construction, we will use additional mechanisms to ensure that the PRG computations done by the honest servers are correct.

[7] Privacy with knowledge of outputs [IKP10] is weaker than security with selective abort and allows the adversary to determine the output of the honest parties.

2. A three-round n-party inner protocol (see Definition 6.1) $(\mathsf{Setup}_{\Pi_j}, \Pi_{j,1}, \Pi_{j,2}, \Pi_{j,3}, \mathsf{out}_{\Pi_j})$ that computes $\mathsf{Eval}(j, \cdot)$ for each $j \in [m]$. From Proposition 6.2, such a protocol can be constructed making black-box use of two-round malicious secure oblivious transfer.

3. A two-round malicious-secure two-party computation protocol $(\mathsf{CRSGen}, \Phi_1, \Phi_2, \mathsf{out}_\Phi)$ for computing the $\mathsf{Sel}_{\lambda n,m}$ functionality. The $\mathsf{Sel}_{\lambda n,m}$ functionality takes in a string ρ_1 from the receiver, $(\rho_2, (s_1, \ldots, s_m))$ from the sender. It computes $\rho_1 \oplus \rho_2$ and uses it as random tape to select a random multiset (with replacement) K of $[m]$ of size λn. It then outputs $(K, \{s_i\}_{i \in K})$ to the receiver. [IKO+11] gave a two-round black-box protocol for computing $\mathsf{Sel}_{\lambda n,m}$ based on two-round malicious-secure OT protocol.

The key lemma that we prove in this section that directly implies Theorem 7.1 is:

Lemma 7.2. *Assuming black-box access to a PRG and to the protocols $\{\Pi_j\}_{j \in [m]}$ and Φ as described above. Then, there exists a three-round protocol for computing any n-party functionality f that satisfies security with unanimous abort against malicious adversaries corrupting an arbitrary subset of the parties.*

As a first step, we construct a protocol over broadcast channels that achieves the weaker notion of security with selective abort.[8] Later, in Remark 7.8, we show how to modify the protocol to achieve the stronger notion of security with unanimous abort.

7.2 Construction

We give the formal description of the construction in Figure 8.

7.3 Proof of Security

Let $\mathcal{A}$ denote the adversary against the protocol and let M denote the set of corrupted parties. Let $H = [n] \setminus M$.

Description of Simulator. Sim does the following:

- **CRS Generation:**
  1. For each $i \in H$ and $k \in M$, it chooses a uniform multiset $K^{k,i}$ of $[m]$ of size λn. Let $K = \cup_{i \in H, k \in M} K^{k,i}$. Note that $|K| \leq \lambda n^3$.
  2. For each $j \in K$, it samples $\mathsf{crs}_j \leftarrow \mathsf{Setup}_{\Pi_j}(1^\lambda)$.
  3. For each $j \in [m] \setminus K$, it samples $(\mathsf{crs}_j, \mathsf{td}_j, \{\pi^i_{j,1}\}_{i \in H}) \leftarrow \mathsf{Sim}_{\Pi_j}(1^\lambda, M)$.
  4. For each $i \in H$ and $k \in M$, it samples $(\mathsf{crs}_{i,k}, \mathsf{td}_{i,k}, \pi_1^{i,k}) \leftarrow \mathsf{Sim}_\Phi(1^\lambda, S)$.
  5. For each $i \in H$ and $k \in M$, it samples $(\mathsf{crs}_{k,i}, \mathsf{td}_{k,i}) \leftarrow \mathsf{Sim}_\Phi(1^\lambda, R)$.
  6. For each $i, k \in H$ and $i, k \in M$, it samples $\mathsf{crs}_{i,k} \leftarrow \mathsf{CRSGen}(1^\lambda)$.
  7. It outputs $(\{\mathsf{crs}_j\}_{j \in [m]}, \{\mathsf{crs}_{i,k}\}_{(i,k) \in [n] \times [n]})$.

[8] In the security with selective abort, the adversary can choose which honest party receives the output and which honest party aborts.

- **CRS Generation:** To generate the CRS,
  1. Sample $\mathsf{crs}_j \leftarrow \mathsf{Setup}_{\Pi_j}(1^\lambda)$ for each $j \in [m]$.
  2. Sample $\mathsf{crs}_{i,k} \leftarrow \mathsf{CRSGen}(1^\lambda)$ for each ordered pair $(i,k) \in [n] \times [n]$.
  3. Output $(\{\mathsf{crs}_j\}_{j\in[m]}, \{\mathsf{crs}_{i,k}\}_{(i,k)\in[n]\times[n]})$.
- **Round-1:** In the first round, each party P_i for $i \in [n]$ does the following:
  1. It samples a uniform MAC key k_i.
  2. It computes $(x_1^i, \ldots, x_m^i) \leftarrow \mathsf{Share}(1^\lambda, i, (x_i, k_i))$.
  3. For each $j \in [m]$, it samples a uniform random string r_j^i and computes $\pi_{j,1}^i \leftarrow \Pi_{j,1}(\mathsf{crs}_j, i, x_j^i; r_j^i)$.
  4. For each $k \in [n] \setminus \{i\}$, it samples a uniform random string $\rho_1^{i,k} \leftarrow \{0,1\}^*$ and computes $\phi_1^{i,k} \leftarrow \Phi_1(\mathsf{crs}_{i,k}, i, \rho_1^{i,k})$.
  5. It broadcasts $\{\pi_{j,1}^i\}_{j\in[m]}$ and $\{\phi_1^{i,k}\}_{k\in[n]\setminus i}$.
- **Round-2:** In the second round, each party P_i for $i \in [n]$ does the following:
  1. It sets $\pi_j(1) = \{\pi_{j,1}^i\}_{i\in[n]}$ for each $j \in [m]$.
  2. For each $j \in [m]$, it computes $\pi_{j,2}^i \leftarrow \Pi_{j,2}(\mathsf{crs}_j, i, x_j^i, \pi_j(1); r_j^i)$.
  3. For each $j \in [m]$, it sets $s_j^{k,i} = (x_j^i, r_j^i)$ for each $k \in [n] \setminus \{i\}$.
  4. For each $k \in [n] \setminus \{i\}$, it samples a uniform random string $\rho_2^{k,i} \leftarrow \{0,1\}^*$ and computes $\phi_2^{k,i} \leftarrow \Phi_2(\mathsf{crs}_{k,i}, i, \phi_1^{k,i}, (\rho_2^{k,i}, (s_1^{k,i}, \ldots, \mathsf{s}_m^{k,i})))$.
  5. It broadcasts $\{\pi_{j,2}^i\}_{j\in[m]}$ and $\{\phi_2^{k,i}\}_{k\in[n]\setminus\{i\}}$.
- **Round-3:** In the third round, each party P_i for $i \in [n]$ does the following:
  1. It sets $\pi_j(2) = \pi_j(1) \cup \{\pi_{j,2}^i\}_{i\in[n]}$ for each $j \in [m]$.
  2. For each $k \in [n] \setminus \{i\}$, it computes $(K^{i,k}, \{s_j^{i,k}\}_{j\in K^{i,k}})$ using out_Φ on $\mathsf{crs}_{i,k}$, $\phi_2^{i,k}$, and the random tape used to generate $\phi_1^{i,k}$.
  3. For each $j \in K^{i,k}$ and for each $k \in [n] \setminus \{i\}$, it:
     (a) Parses $s_j^{i,k}$ as (x_j^k, r_j^k).
     (b) Checks if (x_j^k, r_j^k) is a consistent input, randomness pair that explains the messages sent by P_k in $\pi_j(2)$.
     (c) Checks if the PRG computations in x_j^k are correct.
  4. If any of the above checks fail, then P_i aborts and outputs $\perp$.
  5. Else, it generates $\pi_{j,3}^i \leftarrow \Pi_{j,3}(\mathsf{crs}_j, i, x_j^i, \pi_j(2); r_j^i)$ for each $j \in [m]$.
  6. It broadcasts $\{\pi_{j,3}^i\}_{j\in[m]}$.
- **Output Computation:** To compute the output, P_i for $i \in [n]$ does the following:
  1. It sets $\pi_j(3) = \pi_j(2) \cup \{\pi_{j,3}^i\}_{i\in[n]}$ for each $j \in [m]$.
  2. If any party had aborted before sending its last round message, it outputs $\perp$.
  3. For each $j \in [m]$, it computes $z_j := \mathsf{out}_{\Pi_j}(\mathsf{crs}_j, \pi_j(3))$.
  4. It outputs $(y, \sigma_1, \ldots, \sigma_n) = \mathsf{Dec}(z_1, \ldots, z_m)$.
  5. It checks if $\sigma_i = \mathsf{MAC}(k_i, y)$ and if it is the case, it outputs y and otherwise, it outputs $\perp$.

Figure 8: Black-Box Three-Round MPC Protocol

- **Round-1:**

  1. It runs Sim_Ψ by corrupting the set of clients indexed by M and the set of servers indexed by K. Sim_Ψ provides $\{x_j^i\}_{i\in H, j\in K}$ as the first round message from the honest clients to the corrupt servers.
  2. For each $j \in K$, Sim generates $\pi_{j,1}^i \leftarrow \Pi_{j,1}(\mathsf{crs}_j, i, x_j^i; r_j^i)$ for each $i \in H$.
  3. For each $i \in H$ and $k \in H \setminus \{i\}$, it samples a uniform random string $\rho_1^{i,k} \leftarrow \{0,1\}^*$ and computes $\phi_1^{i,k} \leftarrow \Phi_1(\mathsf{crs}_{i,k}, i, \rho_1^{i,k})$.
  4. It sends $\{\pi_{j,1}^i\}_{j\in[m]}$ and $\{\phi_1^{i,k}\}_{k\in[n]\setminus i}$ for each $i \in H$ to $\mathcal{A}$.
  5. It receives the first round message from $\mathcal{A}$.

- **Round-2:**

  1. It sets $\pi_j(1) = \{\pi_{j,1}^i\}_{i\in[n]}$ for each $j \in [m]$.
  2. For each $i \in H$, $k \in M$ and for each $j \in K$, it sets $s_j^{k,i} = (x_j^i, r_j^i)$. It generates $\phi_2^{k,i} \leftarrow \mathsf{Sim}_\Phi(R, (K^{k,i}, \{s_j^{k,i}\}_{j\in K^{k,i}}))$.
  3. For each $i, k \in H$, it generates $\phi_2^{k,i}$ as $\Phi_2(\mathsf{crs}_{k,i}, i, \phi_1^{k,i}, (\rho_2^{k,i}, (\bot, \ldots, \bot)))$ (for uniformly chosen $\rho_2^{k,i}$).
  4. For each $j \in K$, it generates $\pi_{j,2}^i \leftarrow \Pi_{j,2}(\mathsf{crs}_j, i, x_j^i, \pi_j(1); r_j^i)$ for each $i \in H$.
  5. For each $j \in [m] \setminus K$, it generates $\{\pi_{j,2}^i\}_{i\in H}$ as $\mathsf{Sim}_{\Pi_j}(\mathsf{td}_j, \pi_j(1))$.
  6. It sends $\{\pi_{j,2}^i\}_{j\in[m]}$ and $\{\phi_2^{k,i}\}_{k\in[n]\setminus\{i\}}$ for each $i \in H$ to $\mathcal{A}$.
  7. It receives the second round message from $\mathcal{A}$.

- **Round-3:**

  1. It sets $\pi_j(2) = \pi_j(1) \cup \{\pi_{j,2}^i\}_{i\in[n]}$ for each $j \in [m]$.
  2. For each $i \in H$ and $k \in K$, it runs $\mathsf{Ext}_\Phi(S, \phi_2^{i,k}, \mathsf{td}_{i,k})$ to obtain $(\rho_2^{i,k}, s_1^{i,k}, \ldots, s_m^{i,k})$. It parses $s_j^{i,k}$ as $(x_j^{i,k}, r_j^{i,k})$ for each $j \in M$.
  3. It performs the same checks done by honest P_i by choosing a uniform multiset $K^{i,k}$ of $[m]$ with size λn for each $k \in M$. If any of these checks fail, Sim instructs the ideal functionality to deliver $\bot$ to honest P_i.
  4. It initializes an empty set C_k for each $k \in M$.
  5. For each $j \in [m] \setminus K$:
     (a) It checks if there exists at least one $i \in H$ such that
        i. $(x_j^{i,k}, r_j^{i,k})$ is a consistent input, randomness pair that explains the messages sent by P_k in $\pi_j(2)$, and
        ii. the PRG computations in $x_j^{i,k}$ are correct.
     (b) If no such i exists, then it adds j to C_k.
  6. If there exists at least one $k \in M$ such that $|C_k| \geq \lambda n^2$, then it asks the ideal functionality to send $\bot$ to all the honest parties.

7. It sets $C = \cup_{k \in M} C_k$. Note that $|C| \leq \lambda n^3$. We ask Sim_Ψ to adaptively corrupt the set of servers indexed by C. It provides $\{x_j^i\}_{i \in H, j \in C}$ as the first round message from honest clients to the corrupted servers indexed by C.
8. For each $j \in [m] \setminus (C \cup K)$, for each $k \in M$, it sets $(x_j^k, r_j^k) = (x_j^{i,k}, r_j^{i,k})$ for an $i \in H$ which prevented j from being added to C_k. It provides $\{x_j^k\}_{k \in M, j \in [m] \setminus (C \cup K)}$ to Sim_Ψ as the first round message from the corrupted clients to the honest servers. Sim_Ψ queries the ideal functionality g on $\{(x_i, k_i)\}_{i \in M}$. It queries its ideal functionality on $\{x_i\}_{i \in M}$ and obtains y. For each $i \in H$, it samples σ_i uniformly and for each $i \in M$, it computes $\sigma_i = \mathsf{MAC}(k_i, y)$. It provides the result $(y, \sigma_1, \ldots, \sigma_n)$ to Sim_Ψ who outputs $\{z_j\}_{j \in [m] \setminus (K \cup C)}$ as the second round message from the honest servers. For each $j \in C$, it runs $\mathsf{Sim}_{\Pi_j}(\pi_j(2), \{x_j^k, r_j^k\}_{k \in M}, z_j)$ to compute $\{\pi_{j,3}^i\}_{i \in H}$.
9. For each $j \in C$, it runs $\mathsf{Sim}_{\Pi_j}(\mathsf{td}_j, \pi_j(2), \{x_j^k, r_j^k\}_{k \in M}, \{x_j^i\}_{i \in H})$ to compute $\{\pi_{j,3}^i\}_{i \in H}$ where x_j^k and r_j^k are arbitrarily chosen.
10. It sends $\{\pi_{j,3}^i\}_{j \in [m]}$ for each $i \in H$ to $\mathcal{A}$.

- **Output Computation.**

1. It sets $\pi_j(3) = \pi_j(2) \cup \{\pi_{j,3}^i\}_{i \in [n]}$ for each $j \in [m]$.
2. If any party had aborted before sending its last round message, output $\perp$.
3. For each $j \in [m]$, it computes $z_j := \mathsf{out}_{\Pi_j}(\mathsf{crs}_j, \pi_j(3))$.
4. It computes $(y', \sigma_1', \ldots, \sigma_n') = \mathsf{Dec}(z_1, \ldots, z_m)$.
5. For each $i \in H$, it checks if $(y, \sigma_i) = (y', \sigma_i')$ and if it is the case, it instructs the ideal functionality to send y to P_i and otherwise, it instructs it to send $\perp$.

Proof of Indistinguishability.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the view of the adversary $\mathcal{A}$ and the output of the honest parties in the real execution of the protocol.

- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we make the following changes:

1. For each $i, k \in H \times H$, we generate $\phi_2^{k,i}$ as $\Phi_2(\mathsf{crs}_{k,i}, i, \phi_1^{k,i}, (\rho_2^{k,i}, (\perp, \ldots, \perp)))$.
2. We do not perform the checks described in Round-3 of the protocol for each $k \in H \setminus \{i\}$.

It follows directly from the semi-honest security of the protocol Φ that $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$.

- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we make the following changes:

1. For each $i \in H$ and $k \in M$,
   (a) We generate $(\mathsf{crs}_{i,k}, \mathsf{td}_{i,k}, \pi_1^{i,k}) \leftarrow \mathsf{Sim}_\Phi(1^\lambda, S)$.
   (b) On receiving, $\phi_2^{i,k}$ from $\mathcal{A}$, we run $\mathsf{Ext}_\Phi(S, \phi_2^{i,k}, \mathsf{td}_{i,k})$ to obtain $(\rho_2^{i,k}, s_1^{i,k}, \ldots, s_m^{i,k})$.
   (c) We sample a uniform string $\rho^{i,k}$ and use it to sample a multiset $K^{i,k}$ of $[m]$ of size λn.
   (d) We use $(K^{i,k}, \{s_j^{i,k}\}_{j \in K^{i,k}})$ to perform the checks described in round-3 of the protocol.

In Lemma 7.3, we show that assuming the security of Φ against malicious senders, we have $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$.

- $\underline{\mathsf{Hyb}_3}$: In this hybrid, we make the following changes:

  1. For each $i \in H$ and $k \in M$,
     (a) We generate $(\mathsf{crs}_{k,i}, \mathsf{td}_{k,i}) \leftarrow \mathsf{Sim}_\Phi(1^\lambda, R)$.
     (b) We obtain $\pi_1^{k,i}$ from $\mathcal{A}$ and we run $\mathsf{Ext}_\Phi(R, \phi_1^{k,i}, \mathsf{td}_{k,i})$ to obtain $\rho_1^{k,i}$.
     (c) We sample a multiset $K^{k,i}$ uniformly from $[m]$ of size λn.
     (d) We generate $\phi_2^{k,i} \leftarrow \mathsf{Sim}_\Phi(R, (K^{k,i}, \{s_j^{k,i}\}_{j \in K^{k,i}}))$.

  In Lemma 7.4, we show that assuming the security of Φ against malicious receivers, we have $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$.

- $\underline{\mathsf{Hyb}_4}$: In this hybrid, we make the following changes:

  1. For each $i \in H$ and $k \in M$, we choose a uniform multiset $K^{k,i}$ of $[m]$ of size λn. Let $K = \cup_{i \in H, k \in M} K^{k,i}$. Note that $|K| \leq \lambda n^3$.
  2. For each $k \in M$, $i \in H$ and $j \in M$, we parse $s_j^{i,k}$ as $(x_j^{i,k}, r_j^{i,k})$.
  3. We initialize an empty set C_k for each $k \in M$.
  4. For each $j \in [m] \setminus K$:
     (a) We check if there exists at least one $i \in H$ such that
        i. $(x_j^{i,k}, r_j^{i,k})$ is a consistent input, randomness pair that explains the messages sent by P_k in $\pi_j(2)$, and
        ii. the PRG computations in $x_j^{i,k}$ are correct.
     (b) If no such i exists, then we add j to C_k.
  5. If there exists at least one $k \in M$ such that $|C_k| \geq \lambda n^2$, then we instruct all the honest parties to abort before they send the third round message.

  In Lemma 7.5, we show that $\mathsf{Hyb}_3 \approx_s \mathsf{Hyb}_4$.

- $\underline{\mathsf{Hyb}_5}$: In this hybrid, we make the following changes:

  1. We first generate the set K as before.
  2. For each $j \in [m] \setminus K$, we sample $(\mathsf{crs}_j, \mathsf{td}_j, \{\pi_{j,1}^i\}_{i \in H}) \leftarrow \mathsf{Sim}_{\Pi_j}(1^\lambda, M)$.
  3. On receiving the first round message from $\mathcal{A}$, we set $\pi_j(1)$ as before for each $j \in [m]$. For each $j \in [m] \setminus K$, we generate $\{\pi_{j,2}^i\}_{i \in H}$ as $\mathsf{Sim}_{\Pi_j}(\mathsf{td}_j, \pi_j(1))$.
  4. On receiving the second round message from $\mathcal{A}$, we compute $s_j^{i,k}$ for each $j \in [m]$ and the set C_k for each $k \in M$ as before. If we have not aborted, then we set $C = \cup_{k \in M} C_k$. Note that $|C| \leq \lambda n^3$.
  5. For each $j \in C$, we run $\mathsf{Sim}_{\Pi_j}(\mathsf{td}_j, \pi_j(2), \{x_j^k, r_j^k\}_{k \in M}, \{x_j^i\}_{i \in H})$ to compute $\{\pi_{j,3}^i\}_{i \in H}$ where x_j^k and r_j^k are arbitrarily chosen for each $k \in M$. For each $j \in [m] \setminus (C \cup K)$, for each $k \in M$, set $(x_j^k, r_j^k) = (x_j^{i,k}, r_j^{i,k})$ for an $i \in H$ which prevented j from being added to C_k. We run $\mathsf{Sim}_{\Pi_j}(\pi_j(2), \{x_j^k, r_j^k\}_{k \in M}, \mathsf{Eval}(j, x_j^1, \ldots, x_j^n))$ to compute $\{\pi_{j,3}^i\}_{i \in H}$.

In Lemma 7.6, we rely on the security of the multiparty inner protocol to show that $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$.

- $\underline{\mathsf{Hyb}_6}$: In this hybrid, we make the following changes:

  1. We generate the set K as before.
  2. We start running the simulator Sim_Ψ for the outer protocol by corrupting the set of clients given by M and the set of servers indexed by K. Sim_Ψ provides $\{x_j^i\}_{i\in H, j\in K}$ as the first round message sent by the honest clients to the corrupted servers. We use this to generate the first two round messages on behalf of the honest parties.
  3. On receiving the second round message from $\mathcal{A}$, we compute the set C as before. We instruct Sim_Ψ to adaptively corrupt the set of servers indexed by C and obtain $\{x_j^i\}_{i\in H, j\in C}$. We use this to generate $\pi_{j,3}^i$ for each $i \in H$ and $j \in C$ as described in step-5 of Hyb_5.
  4. For each $j \in [m] \setminus (C \cup K)$, for each $k \in M$, we compute (x_j^k, r_j^k) as in step-5 of Hyb_5. We send $\{x_j^k\}_{k\in M, j\in[m]\setminus(K\cup C)}$ as the first round message sent by the malicious clients to the honest servers.
  5. Sim_Ψ queries the ideal functionality g on $\{(x_i, k_i)\}_{i\in M}$. We compute the output of g honestly and return the result to Sim_Ψ who outputs $\{z_j\}_{j\in[m]\setminus(K\cup C)}$ as the second round message from the honest servers. We use this to compute $\pi_{j,3}^i$ for each $i \in H$ and $j \in [m] \setminus K \cup C$ as described in step-5 of Hyb_5.

  In Lemma 7.7, we show that $\mathsf{Hyb}_5 \approx_c \mathsf{Hyb}_6$ from the security of the outer protocol Ψ.

- $\underline{\mathsf{Hyb}_7}$: In this hybrid, we make the following changes:

  1. When computing the output of g in step-5 of Hyb_6, we compute $y = f(x_1, \ldots, x_n)$ and for each $i \in H$, we sample σ_i uniformly and for each $i \in M$, we compute σ_i as $\mathsf{MAC}(k_i, y)$. We return $(y, \sigma_1, \ldots, \sigma_n)$ to Sim_Ψ.
  2. In computing the output of each honest P_i, we check if the computed (y', σ_i') in the output phase is exactly same as the above sampled one. If not, we instruct P_i to abort.

  It follows from the uniformity of the tags and the one-time unforgeability of the MAC scheme that $\mathsf{Hyb}_6 \approx_s \mathsf{Hyb}_7$. Note that Hyb_7 is identically distributed to the ideal experiment using Sim.

Lemma 7.3. *Assuming the security of protocol Φ against malicious senders, we have $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$.*

Proof. Assume for the sake of contradiction that Hyb_1 and Hyb_2 are computationally distinguishable. We show that this contradicts the security of Φ against malicious senders.

We start interacting with the external challenger and provide for each $i \in H$ and $k \in M$, a uniformly chosen random string $\rho_1^{i,k}$ as the challenge honest receiver inputs. The challenger responds with $\mathsf{crs}_{i,k}$ and $\pi_1^{i,k}$ for each $i \in H$ and $k \in M$ and we use this to generate the CRS and the first round message of the protocol. On receiving the second round message from $\mathcal{A}$, we send $\{\pi_2^{i,k}\}_{i\in H, k\in M}$ to the external challenger. The challenger responds with $(K^{i,k}, \{s_j^{i,k}\}_{j\in K^{i,k}})$ for each $i \in H$ and $k \in M$ and we use this to perform the same checks as described in round-3 of the protocol. We compute the output of the honest parties as before and output the view of $\mathcal{A}$ and the output of honest parties.

If the view generated by the external challenger was identical to the real execution of the protocol Φ, then the output of the above reduction is identically distributed to Hyb_1. Else, it is distributed identically distributed to Hyb_2. Since Hyb_1 and Hyb_2 are computationally distinguishable with non-negligible advantage, the above reduction breaks the security of Φ against malicious senders with non-negligible advantage and this is a contradiction. □

Lemma 7.4. *Assuming the security of protocol Φ against malicious receivers, we have $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$.*

Proof. Assume for the sake of contradiction that Hyb_2 and Hyb_3 are computationally distinguishable with non-negligible advantage. We show that this contradicts the security of Φ against malicious receivers.

We start interacting with the external challenger and provide for each $i \in H$ and $k \in M$, $(\rho_2^{k,i}, s_1^{k,i}, \ldots, s_m^{k,i})$ (where $\rho_2^{k,i}$ is uniformly chosen) as the challenge sender inputs. The challenger provides $\mathsf{crs}_{k,i}$ for each $k \in M$ and $i \in H$ and we use this to generate the CRS of the overall protocol. We receive $\pi_1^{k,i}$ from $\mathcal{A}$ for each $k \in M$ and $i \in H$ and we forward these messages to the external challenger. The challenger responds with $\{\pi_2^{k,i}\}_{i \in H, k \in M}$ and we use this to generate the second round message of the overall protocol. We compute the output of the honest parties as before and we finally output the view of the adversary $\mathcal{A}$ and the outputs of all the honest parties.

Note that since $\rho_2^{k,i}$ is uniformly chosen, $K^{k,i}$ is distributed as a random multiset of $[m]$ of size λn. If the view generated by the external challenger was identical to the real execution of the protocol Φ, then the output of the above reduction is identically distributed to Hyb_2. Else, it is distributed identically distributed to Hyb_3. Since Hyb_2 and Hyb_3 are computationally distinguishable with non-negligible advantage, the above reduction breaks the security of Φ against malicious receivers with non-negligible advantage and this is a contradiction. □

Lemma 7.5. $\mathsf{Hyb}_3 \approx_s \mathsf{Hyb}_4$.

Proof. Note that the only difference between Hyb_3 and Hyb_4 is that in Hyb_4 we abort if there exists a $k \in M$ such that $|C_k| \geq \lambda n^2$. To show that Hyb_3 and Hyb_4 are statistically close, we prove that if the above condition holds, then in Hyb_3, the checks done by each honest P_i (for $i \in H$) fails with overwhelming probability.

Fix some honest P_i where $i \in H$. Note that $K^{i,k}$ is distributed as a random multiset of $[m]$ of size λn. If $C_k \cap K^{i,k} \neq \emptyset$, then the the honest P_i in Hyb_3 also aborts. We show that this event happens with overwhelming probability.

$$\begin{aligned}
\Pr[|K^{i,k} \cap C_k| = 0] &= (1 - \frac{|C_k|}{m})^{\lambda n} \\
&\leq e^{-|C_k|\lambda n/m} \\
&\leq e^{-\lambda^2 n^3/m} \\
&\leq e^{-O(\lambda)}
\end{aligned}$$

where the last inequality follows since $m = O(\lambda n^3)$. The lemma now follows from a union bound over each $i \in H$. □

Lemma 7.6. *Assuming the security of the multiparty inner protocol Π_j for each $j \in [m]$, we have $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$.*

Proof. Assume for the sake of contradiction that Hyb_4 and Hyb_5 are computationally distinguishable. We show that this contradicts the security of the multiparty inner protocol.

For each $i \in H$ and $k \in M$, we choose a uniform multiset $K^{k,i}$ of $[m]$ of size λn. Let $K = \cup_{i \in H, k \in M} K^{k,i}$. Note that $|K| \leq \lambda n^3$. Let $\prec$ denote a total ordering on the elements in $[m] \setminus K$. Since Hyb_4 and Hyb_5 are computationally distinguishable, by standard averaging argument, there exists Hyb'_4 and Hyb'_5 (described below) that are computationally distinguishable with non-negligible advantage. There exists a $j \in [m] \setminus K$ such that in both hybrids, for each $j^* \prec j$, the distribution of the messages in Π_{j^*} is identical to Hyb_5 and for each j^* such that $j \prec j^*$, the distribution of the messages in Π_{j^*} is identical to Hyb_4. The only difference is that in Hyb'_4, the messages in Π_j are generated as in Hyb_4 whereas in Hyb'_5, it is generated as in Hyb_5. We now show that this contradicts the security of the multiparty inner protocol Π_j.

We start interacting with the external challenger by providing $\{x^i_j\}_{i \in H}$ as the set of honest party inputs. We receives $(\mathsf{crs}_j, \{\pi^i_{j,1}\}_{i \in H})$ from the challenger and use this to generate the CRS and the first round message in the protocol. We receive the first round message from $\mathcal{A}$ and we forward $\{\pi^i_{j,1}\}_{i \in M}$ to the external challenger. The challenger responds with $\{\pi^i_{j,2}\}_{i \in H}$. We use this to generate the second round message in the overall protocol. On receiving the second round message from $\mathcal{A}$, we compute $s^{i,k}_j$ for each $j \in [m]$ and the set C_k for each $k \in M$ as before. If we have not aborted, then we set $C = \cup_{k \in M} C_k$. If $j \in C$, we send $\{\pi^k_{j,2}, x^k_j, r^k_j\}_{k \in M}$ to the challenger where x^k_j and r^k_j are arbitrarily chosen for each $k \in M$. Else, for each $k \in M$, we set $(x^k_j, r^k_j) = (x^{i,k}_j, r^{i,k}_j)$ for an $i \in H$ which prevented j from being added to C_k. We send $\{\pi^k_{j,2}, x^k_j, r^k_j\}_{k \in M}$ to the external challenger. The external challenger responds with $\{\pi^i_{j,3}\}_{i \in H}$ and we use this compute the output of all the honest parties in H as before. We finally output the view of $\mathcal{A}$ and the outputs of all the honest parties.

If the view generated by the external challenger was identical to the real execution of the protocol Π_j, then the output of the above reduction is identically distributed to Hyb'_4. Else, it is distributed identically distributed to Hyb'_5. Since Hyb'_4 and Hyb'_5 are computationally distinguishable with non-negligible advantage, the above reduction breaks the security of multiparty inner protocol with non-negligible advantage and this is a contradiction. □

Lemma 7.7. *Assuming the privacy with knowledge of outputs property of the outer protocol Ψ, we have $\mathsf{Hyb}_5 \approx_c \mathsf{Hyb}_6$.*

Proof. Assume for the sake of contradiction that Hyb_5 and Hyb_6 are computationally distinguishable with non-negligible advantage. We show that this contradicts the security of the outer protocol Ψ.

We start interacting with the external challenger and provide $\{(x_i, k_i)\}_{i \in H}$ where k_i is uniformly chosen MAC key as the honest party inputs. We instruct the challenger to corrupt the set of servers indexed by K and the set of clients indexed by M. The challenger responds with $\{x^i_j\}_{i \in H, j \in K}$ and we use it to generate the first two messages of the protocol. On receiving the second round message from $\mathcal{A}$, we compute the set C as before. If $|C_k| < \lambda n^2$ for each $k \in M$, we instruct Sim_Ψ to adaptively corrupt the set of servers indexed by C and obtain $\{x^i_j\}_{i \in H, j \in C}$. We use this to generate $\pi^i_{j,3}$ for each $i \in H$ and $j \in C$ as described in step-5 of Hyb_5. For each $j \in [m] \setminus (C \cup K)$, for each $k \in M$, we compute x^k_j, r^k_j as in step-5 of Hyb_5. We send $\{x^k_j\}_{k \in M, j \in [m] \setminus (K \cup C)}$ as the first round message sent by the malicious clients to the honest servers. The challenger returns $\{z_j\}_{j \in [m] \setminus (K \cup C)}$ as the second round message from the honest servers. We use this to compute $\pi^i_{j,3}$ for each $i \in H$ and $j \in [m] \setminus K \cup C$ as described in simulation. We compute the output of the honest parties as before and finally output the view of $\mathcal{A}$ and the output of all the honest parties.

If the view generated by the external challenger was identical to the real execution of the protocol Ψ, then the output of the above reduction is identically distributed to Hyb_5. Else, it is distributed identically distributed to Hyb_6. Since Hyb_5 and Hyb_6 are computationally distinguishable with non-negligible advantage, the above reduction breaks the privacy with knowledge of outputs property of Ψ with non-negligible advantage and this is a contradiction. □

Remark 7.8 (On achieving Security with Unanimous Abort)**.** *We note that if we replace* MAC *with an one-time digital signature scheme then we can get security with unanimous abort. Further, note that the signing algorithm of Lamport's one-time signature scheme does not involve any cryptographic operations and hence, we still continue to make black-box use of cryptographic primitives.*

Acknowledgments. Y. Ishai was supported in part by ERC Project NTSC (742754), BSF grant 2018393, and ISF grant 2774/20. D. Khurana was supported in part by DARPA SIEVE award, a gift from Visa Research, and a C3AI DTI award. A. Sahai was supported in part from a Simons Investigator Award, DARPA SIEVE award, NTT Research, NSF Frontier Award 1413955, BSF grant 2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through Award HR00112020024. A. Srinivasan was supported in part by a SERB startup grant.

References

[ABG+20] Benny Applebaum, Zvika Brakerski, Sanjam Garg, Yuval Ishai, and Akshayaram Srinivasan. Separating two-round secure computation from oblivious transfer. In *ITCS 2020*, volume 151 of *LIPIcs*, pages 71:1–71:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[ABG+21] Amit Agarwal, James Bartusek, Vipul Goyal, Dakshita Khurana, and Giulio Malavolta. Two-round maliciously secure computation with super-polynomial simulation. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part I*, volume 13042 of *Lecture Notes in Computer Science*, pages 654–685. Springer, 2021.

[ACJ17] Prabhanjan Ananth, Arka Rai Choudhuri, and Abhishek Jain. A new approach to round-optimal secure multiparty computation. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 468–499, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[AIK05] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*, pages 260–274. IEEE Computer Society, 2005.

[AIR01] William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 119–135, Innsbruck, Austria, May 6–10, 2001. Springer, Heidelberg, Germany.

[AMR21] Behzad Abdolmaleki, Giulio Malavolta, and Ahmadreza Rahimi. Two-round concurrently secure two-party computation. *IACR Cryptol. ePrint Arch.*, page 1357, 2021.

[App17] Benny Applebaum. Garbled circuits as randomized encodings of functions: a primer. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 1–44. Springer International Publishing, 2017.

[BCG+19] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In *CCS 2019*, pages 291–308. ACM, 2019.

[BDM22] Pedro Branco, Nico Döttling, and Paulo Mateus. Two-round oblivious linear evaluation from learning with errors. In *PKC 2022, Part I*, pages 379–408, 2022.

[Bea95] Donald Beaver. Precomputing oblivious transfer. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 97–109, Santa Barbara, CA, USA, August 27–31, 1995. Springer, Heidelberg, Germany.

[BF22] Nir Bitansky and Sapir Freizeit. Statistically sender-private OT from LPN and derandomization. In *Crypto 2022*, 2022.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC 1988*, pages 103–112, 1988.

[BGI+17] Saikrishna Badrinarayanan, Sanjam Garg, Yuval Ishai, Amit Sahai, and Akshay Wadia. Two-message witness indistinguishability and secure computation in the plain model from new assumptions. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 275–303, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany.

[BGI+18] Elette Boyle, Niv Gilboa, Yuval Ishai, Huijia Lin, and Stefano Tessaro. Foundations of homomorphic secret sharing. In *ITCS 2018*, pages 21:1–21:21, January 2018.

[BGJ+17] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Dakshita Khurana, and Amit Sahai. Round optimal concurrent MPC via strong simulation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 743–775, Baltimore, MD, USA, November 12–15, 2017. 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. LNCS, pages 459–487, Santa Barbara, CA, USA, 2018. Springer, Heidelberg, Germany.

[BHP17] Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 645–677, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 12*, pages 784–796, Raleigh, NC, USA, October 16–18, 2012. ACM Press.

[BL18] Fabrice Benhamouda and Huijia Lin. k-round MPC from k-round OT via garbled interactive circuits. *EUROCRYPT*, 2018.

[BLV03] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower bounds for non-black-box zero knowledge. In *FOCS 2003*, pages 384–393, 2003.

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

[Can20] Ran Canetti. Universally composable security. *J. ACM*, 67(5):28:1–28:94, 2020.

[CCG+20] Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Round optimal secure multiparty computation from minimal assumptions. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, pages 291–319, 2020.

[CDI+19] Melissa Chase, Yevgeniy Dodis, Yuval Ishai, Daniel Kraschewski, Tianren Liu, Rafail Ostrovsky, and Vinod Vaikuntanathan. Reusable non-interactive secure computation. LNCS, pages 462–488, Santa Barbara, CA, USA, 2019. Springer, Heidelberg, Germany.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In John H. Reif, editor, *STOC 2002*, pages 494–503, 2002.

[DGH+20] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, Daniel Masny, and Daniel Wichs. Two-round oblivious transfer from CDH or LPN. In *EUROCRYPT 2020, Part II*, pages 768–797, 2020.

[DIO21] Samuel Dittmer, Yuval Ishai, and Rafail Ostrovsky. Line-point zero knowledge and its applications. In *ITC 2021*, pages 5:1–5:24, 2021.

[FJK21] Rex Fernando, Aayush Jain, and Ilan Komargodski. Maliciously-secure mrnisc in the plain model. *IACR Cryptol. ePrint Arch.*, page 1319, 2021.

[GGJS12] Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Concurrently secure computation in constant rounds. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 99–116, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[GIKR02] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. On 2-round secure multiparty computation. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 178–193, 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 *TCC 2018, Part I*, LNCS, pages 123–151. Springer, Heidelberg, Germany, March 2018.

[GKP17] Sanjam Garg, Susumu Kiyoshima, and Omkant Pandey. On the exact round complexity of self-composable two-party computation. In Jean-Sébastien Coron and Jesper Buus

Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 194–224, Paris, France, May 8–12, 2017. Springer, Heidelberg, Germany.

[GMPP16] Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 448–476, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 695–704, San Jose, CA, USA, June 6–8, 2011. ACM Press.

[GS18] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. LNCS, pages 468–499. Springer, Heidelberg, Germany, 2018.

[HHPV18] Shai Halevi, Carmit Hazay, Antigoni Polychroniadou, and Muthuramakrishnan Venkitasubramaniam. Round-optimal secure multi-party computation. LNCS, pages 488–520, Santa Barbara, CA, USA, 2018. Springer, Heidelberg, Germany.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

[IKO+11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 406–425, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.

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

[IKSS21] Yuval Ishai, Dakshita Khurana, Amit Sahai, and Akshayaram Srinivasan. On the round complexity of black-box secure MPC. In *CRYPTO 2021*, volume 12826 of *Lecture Notes in Computer Science*, pages 214–243. Springer, 2021.

[IKSS22] Yuval Ishai, Dakshita Khurana, Amit Sahai, and Akshayaram Srinivasan. Round-optimal black-box protocol compilers. In *EUROCRYPT*, 2022.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.

[IR90] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 8–26, Santa Barbara, CA, USA, August 21–25, 1990. Springer, Heidelberg, Germany.

[KMO14] Susumu Kiyoshima, Yoshifumi Manabe, and Tatsuaki Okamoto. Constant-round black-box construction of composable multi-party computation protocol. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 343–367, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany.

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 335–354, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany.

[KOS03] Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Round efficiency of multi-party computation with a dishonest majority. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 578–595, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany.

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009.

[MR19] Daniel Masny and Peter Rindal. Endemic oblivious transfer. In *CCS 2019*, pages 309–326. ACM, 2019.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA.*, pages 448–457. ACM/SIAM, 2001.

[ORS15] Rafail Ostrovsky, Silas Richelson, and Alessandra Scafuro. Round-optimal black-box two-party computation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 339–358, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[Pas12] Anat Paskin-Cherniavsky. *Secure Computation with Minimal Interaction*. PhD thesis, Technion, 2012. Available at http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/2012/PHD/PHD-2012-16.pdf.

[PS21] Arpita Patra and Akshayaram Srinivasan. Three-round secure multiparty computation from black-box two-round oblivious transfer. In *CRYPTO 2021*, volume 12826 of *Lecture Notes in Computer Science*, pages 185–213. Springer, 2021.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 1–20, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany.

[Wee10] Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *51st FOCS*, pages 531–540, Las Vegas, NV, USA, October 23–26, 2010. IEEE Computer Society Press.

[Yao86a] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167. IEEE Computer Society, 1986.

[Yao86b] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press.

A Proof of Proposition 4.3

In this section, we give the proof of proposition 4.3.

A.1 Construction

Let $f : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}^*$ be the two-party functionality to be computed. Let $x_0 \in \{0,1\}^m$ be the private input of P_0 and $x_1 \in \{0,1\}^m$ be the private input of P_1. Let $(\mathsf{CRSGen}, \mathsf{OT}_1, \mathsf{OT}_2, \mathsf{out}_{\mathsf{OT}})$ be a two-round malicious-secure oblivious transfer protocol. Let $(\mathsf{Garble}, \mathsf{Eval})$ be a circuit garbling scheme. We give the description of the construction in Figure 9. This construction is exactly the same as Yao's protocol [Yao86b] except for the usage of a malicious-secure OT protocol.

- **CRS Generation:** To generate the CRS,
  1. $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda)$.
  2. Output crs.
- $\Pi_1(\mathsf{crs}, x_0)$:
  1. For each $j \in [m]$, compute $\mathsf{otr}_j \leftarrow \mathsf{OT}_1(\mathsf{crs}, x_{0,j}; r_{0,j})$ where $r_{0,j}$ is uniformly chosen.
  2. Output $\pi_1 = \{\mathsf{otr}_j\}_{j\in[m]}$.
- $\Pi_2(\mathsf{crs}, x_1, \pi_1)$:
  1. Compute $\widetilde{f}, \{\mathsf{lab}^0_{j,1}, \mathsf{lab}^0_{j,1}\}_{j\in[m]}, \{\mathsf{lab}^1_{j,1}, \mathsf{lab}^1_{j,1}\}_{j\in[m]} \leftarrow \mathsf{Garble}(1^\lambda, f)$.
  2. For each $j \in [m]$, compute $\mathsf{ots}_j \leftarrow \mathsf{OT}_2(\mathsf{crs}, \mathsf{otr}_j, \{\mathsf{lab}^0_{j,0}, \mathsf{lab}^0_{j,1}\})$.
  3. Output $\pi_2 = (\widetilde{f}, \{\mathsf{otr}_j, \mathsf{lab}^1_{j,x_{1,j}}\}_{j\in[m]})$.
- $\mathsf{out}_\Pi(\mathsf{crs}, \pi_2, (x_0, \{r_{0,j}\}_{j\in[m]}))$**:**
  1. For each $j \in [m]$, compute $\mathsf{lab}^0_{j,x_{0,j}} \leftarrow \mathsf{out}_{\mathsf{OT}}(\mathsf{crs}, \mathsf{ots}_j, (x_{0,j}, r_{0,j}))$.
  2. It outputs $\mathsf{Eval}(\widetilde{f}, \{\mathsf{lab}^0_{j,x_{0,j}}, \mathsf{lab}^1_{j,x_{1,j}}\}_{j\in[m]})$.

Figure 9: Construction of Inner Protocol with Output Equivocality

A.2 Proof of Security

In this subsection, we show that the construction described in Figure 9 satisfies Definition 4.1.

A.2.1 Security against Malicious Receivers

Let $\mathcal{A}$ be an adversary that is corrupting the receiver.

Description of Simulator. Sim_Π runs $\mathsf{Sim}_{\mathsf{OT}}$ to generate the $(\mathsf{crs}, \mathsf{td})$. On receiving $\{\mathsf{otr}_j\}_{j\in[m]}$ from $\mathcal{A}$, Ext_Π runs $\mathsf{Ext}_{\mathsf{OT}}(R, \mathsf{otr}_j, \mathsf{td})$ for each $j \in [m]$ to compute $x_{0,j}$. To compute the final round message, it runs the simulator for the garbled circuits $\mathsf{Sim}_{\mathsf{GC}}$ on input $(1^\lambda, 1^{|f|}, 1^{2m}, f(x_0, x_1))$ to compute $\widetilde{f}, \{\mathsf{lab}_j^0, \mathsf{lab}_j^1\}_{j\in[m]}$. It computes $\mathsf{ots}_j \leftarrow \mathsf{OT}_2(\mathsf{crs}, \mathsf{otr}_1, \{\mathsf{lab}_j^0, \mathsf{lab}_j^0\})$. It sends $\widetilde{f}, \{\mathsf{lab}_j^1, \mathsf{ots}_j\}_{j\in[m]}$ to $\mathcal{A}$.

We now that Real_R and Ideal_R described in Figure 3 are computationally indistinguishable.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the output of the experiment $\mathsf{Real}_R(1^\lambda, \mathcal{A}, x_1)$.
- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we make the following changes:

  1. We generate $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Sim}_{\mathsf{OT}}(1^\lambda, S)$.
  2. On receiving $\{\mathsf{otr}_j\}_{j\in[m]}$ from $\mathcal{A}$, we run $\mathsf{Ext}_{\mathsf{OT}}(S, \mathsf{otr}_j, \mathsf{td}_j)$ to obtain $x_{0,j}$ for each $j \in [m]$.
  3. We generate $\mathsf{ots}_j \leftarrow \mathsf{OT}_2(\mathsf{crs}, \{\mathsf{lab}_{x_{0,j}}^0, \mathsf{lab}_{x_{0,j}}^0\})$.

  In Lemma A.1, we rely on the security of the oblivious transfer against malicious receivers to show that $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$.

- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we generate $\widetilde{f}, \{\mathsf{lab}_{j,x_{0,j}}^0, \mathsf{lab}_{j,x_{1,j}}^1\}_{j\in[m]}$ as the output of simulator for garbled circuits $\mathsf{Sim}_{\mathsf{GC}}$ with input $(1^\lambda, 1^{|f|}, 1^{2m}, f(x_0, x_1))$. It now follows directly from the security of garbled circuits that Hyb_1 and Hyb_2 are computationally distinguishable. Note that Hyb_2 is identically distributed to the experiment $\mathsf{Ideal}_R(1^\lambda, \mathcal{A}, x_1, (\mathsf{Sim}_\Pi, \mathsf{Ext}_\Pi))$

Lemma A.1. *Assuming the security of the oblivious transfer protocol against malicious receivers, we have* $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$.

Proof. Assume for the sake of contradiction that Hyb_0 and Hyb_1 are computationally distinguishable with non-negligible advantage. We now show that this contradicts the security of oblivious transfer protocol against malicious receivers.

We start interacting the OT challenger and provide $\{\mathsf{lab}_{j,0}^0, \mathsf{lab}_{j,1}^0\}_{j\in[m]}$ as the set of sender challenge inputs. The challenger provides crs and we use this to start the interaction with $\mathcal{A}$. On receiving $\{\mathsf{otr}_j\}_{j\in[m]}$ from the adversary, we forward this to the challenger and the challenger gives $\{\mathsf{ots}_j\}_{j\in[m]}$. We use this to generate the second round message in the protocol as before.

If the view generated by the challenger corresponds to the real experiment in the OT security game, then the view of view of $\mathcal{A}$ created by the above game corresponds to Hyb_0. Else, it corresponds to Hyb_1. Since Hyb_0 and Hyb_1 are distinguishable with non-negligible advantage, the above reduction breaks the security of the OT protocol against malicious receivers with non-negligible advantage and this is a contradiction. □

A.2.2 Correctness of Extraction

The correctness of extraction property follows directly from the correctness of extraction for the underlying OT protocol which follows from the security property of OT against malicious receivers.

A.2.3 Robust Security against Semi-Malicious Senders

Let $\mathcal{A}$ be an adversary that corrupts the sender.

Description of Simulator. Sim_Π runs $\mathsf{Sim}_{\mathsf{OT}}(1^\lambda)$ to obtain $(\mathsf{crs}, \mathsf{td}, \{\mathsf{otr}_j\}_{j\in[m]})$. It sends $\{\mathsf{otr}_j\}_{j\in[m]}$ to $\mathcal{A}$ and obtains the second round message π_2. Ext_Π uses td and $\{\mathsf{ots}_j\}_{j\in[m]}$ to extract $\{\mathsf{lab}^0_{j,0}, \mathsf{lab}^0_{j,1}\}_{j\in[m]}$. It sets st to be this value. To compute the output, Sim_Π runs $\mathsf{Eval}(\widetilde{f}, \{\mathsf{lab}^0_{j,x_{0,j}}, \mathsf{lab}^1_{j,x_{1,j}}\}_{j\in[m]})$.

Proof of Indistinguishability.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the output of the experiment Real_S described in Figure 4.
- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we make the following changes:

  1. We run $\mathsf{Sim}_{\mathsf{OT}}(1^\lambda, S)$ to obtain $(\mathsf{crs}, \mathsf{td}, \{\mathsf{otr}_j\}_{j\in[m]})$.
  2. On receiving $\{\mathsf{ots}_j\}_{j\in[m]}$ from $\mathcal{A}$, we run $\mathsf{Ext}_{\mathsf{OT}}$ on this and td to obtain $\{\mathsf{lab}^0_{j,0}, \mathsf{lab}^0_{j,1}\}_{j\in[m]}$. We use this and the input x_0 to evaluate the garbled circuit $\widetilde{f}$.

  In Lemma A.2, we use the security of the oblivious transfer against malicious senders to prove that $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$.

- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we make the following changes:

  1. If (x_1, r_1) output by $\mathcal{A}$ is a consistent input, randomness pair w.r.t. to π_2, then we generate the output of P_0 as $f(x_0, x_1)$. Otherwise, we compute the output as in Hyb_1.

  Note that Hyb_2 is identically distributed to Hyb_1 from the perfect correctness of garbled circuits. Observe that Hyb_2 is identically distributed to the output of Ideal_S experiment.

Lemma A.2. *Assuming the security of oblivious transfer protocol against malicious senders, we have* $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$.

Proof. Assume for the sake of contradiction that Hyb_0 and Hyb_1 are computationally distinguishable with non-negligible advantage. We now show that this contradicts the security of oblivious transfer protocol against malicious senders.

We start interacting the OT challenger and provide $\{x_{0,j}\}_{j\in[m]}$ as the set of receiver challenge inputs. The challenger provides crs and $\{\mathsf{otr}_j\}_{j\in[m]}$ and we use this to start the interaction with $\mathcal{A}$. On receiving $\{\mathsf{ots}_j\}_{j\in[m]}$ from the adversary, we forward this to the challenger and the challenger gives $\{\mathsf{lab}^0_{j,x_{0,j}}\}_{j\in[m]}$. We use this to evaluate the garbled circuit $\widetilde{f}$ as before.

If the view generated by the challenger corresponds to the real experiment in the OT security game, then the view of view of $\mathcal{A}$ created by the above game corresponds to Hyb_0. Else, it corresponds to Hyb_1. Since Hyb_0 and Hyb_1 are distinguishable with non-negligible advantage, the above reduction breaks the security of the OT protocol against malicious senders with non-negligible advantage and this is a contradiction. □

B Two-Round Outer MPC Protocol

We recall the following theorem from [IKP10, Pas12].

Theorem B.1 ([IKP10, Pas12])**.** *Let g be computable by polynomial-size branching programs over a finite field. Then, for any $n, t \in \mathbb{N}$, there exists an n-client, $m = (3t+1)$ server protocol for computing g satisfying statistical security with selective abort against an arbitrary number of client corruptions and t server corruptions.*

The construction given below is almost the same as that of pairwise verifiable MPC given in [IKSS22]. We first give a construction based on PRFs and later in Remark B.2, we show how to rely only on a PRG.

Notation. We recall the BMR garbled circuit [BMR90] construction. We slightly modify the BMR garbled gadget to also include one-time MACs. Let f be computed by a Boolean circuit C that comprises entirely of fan-in 2 NAND gates. The BMR garbling gadget comprises of the following components:

1. For each wire w in C, each party $i \in [n]$, chooses a uniform mask bit b_w^i and two random PRF keys $k_{w,0}^i, k_{w,1}^i \leftarrow \{0,1\}^\lambda$. If w is the input wire of some party P_j, then $b_w^i = 0$ and $k_{w,0}^i = 0^\lambda$ and $k_{w,1}^i = 0^\lambda$ for each $i \neq j$. If w is the output wire, then $b_w^i = 0$ and $k_{w,0}^i = 0^\lambda$ and $k_{w,1}^i = 0^\lambda$ for each $i \in [n]$. We use b_w to denote $\bigoplus_{i=1}^n b_w^i$. For each gate g and for each r_1, r_2, the parties also sample a one-time MAC key κ_{g,r_1,r_2}^i.

2. For each NAND gate g whose input wires are x and y and the output wire is z, the garbled gate is given by $\{\widetilde{G}_{r_1,r_2}\}_{r_1,r_2\in\{0,1\}}$ where:

$$\begin{aligned}\widetilde{G}_{r_1,r_2} \quad = \quad & \Big(\bigoplus_{i=1}^{n} F_{k_{x,r_1}^i}(g,r_1,r_2) \oplus \bigoplus_{i=1}^{n} F_{k_{y,r_2}^i}(g,r_1,r_2)\Big) \oplus \\ & \Big(y_{g,r_1,r_2} = \big(\{k_{z,\chi_{g,r_1,r_2}}^i\}_{i\in[n]}, \chi_{g,r_1,r_2}\big), \{\mathsf{MAC}_{\kappa_{r_1,r_2}^i}(y_{g,r_1,r_2})\}_{i\in[n]}\Big)\end{aligned}$$

   where $\chi_{g,r_1,r_2} = b_z \oplus g(r_1 \oplus b_x, r_2 \oplus b_y)$.

3. The parties broadcast $k_{w,x_w\oplus b_w}^i$ and $x_w \oplus b_w$ to every other party and the parties use this to evaluate the BMR garbled gadget just like Yao's garbled evaluation procedure and obtain the output.

We note that $\widetilde{G}_{r_1,r_2}$ and $(k_{w,x_w\oplus b_w}^i, x_w \oplus b_w)$ can be computed by a branching program that takes $x_w, b_w, \{k_{x,r_1}^i, k_{y,r_2}^i, F_{k_{x,r_1}^i}(g,r_1,r_2), F_{k_{x,r_2}^i}(g,r_1,r_2), \kappa_{g,r_1,r_2}^i\}_{i\in[n],x,y,g\in[|C|],r_1,r_2\in\{0,1\}}$ as inputs. Let h be the set of all such computations done in the BMR garbling gadget. We observe that h is computable by a branching program over a finite field.

Description of the Protocol. The protocol for computing the BMR garbled circuit uses the protocol from Theorem B.1 for computing the functionality h. Note that in this protocol, the client pre-computes the PRF functionality and feeds it as input to h. The output computation corresponds to the evaluation of the BMR garbled circuit and checking the correctness of the MACs.

Sketch of Proof of Security. To show security, we consider a sequence of hybrids starting from the real execution and ending with the ideal execution that defines our simulator Sim_Φ. Consider an admissible adversary $\mathcal{A}$ corrupting a subset M of the clients and a subset T of the servers.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the view of the adversary and the outputs of the honest output clients in the real execution of the protocol.
- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we use the simulator for computing h from Theorem B.1 to simulate the view of $\mathcal{A}$ and generate the outputs of all the honest output clients. This hybrid is statistically indistinguishable from the previous one due to the security of protocol for computing h from Theorem B.1.
- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we replace each garbled gate gadget $\{\widetilde{G}_{r_1,r_2}\}_{r_1,r_2\in\{0,1\}}$ in the BMR garbled circuit with the simulated one. This hybrid is computationally indistinguishable from the previous hybrid from the security of the PRF of the honest parties.
- $\underline{\mathsf{Hyb}_3}$: In this hybrid, we extract k^i_{x,r_1} and k^i_{y,r_2} sent by the adversary on behalf of some malicious client i. This extraction is done via the simulator for the protocol for computing h. We then compute $F_{k^i_{x,r_1}}(g,r_1,r_2)$ and $F_{k^i_{y,r_2}}(g,r_1,r_2)$ from these values. We use the simulator for the protocol computing h to extract δ^i_{x,r_1} and δ^i_{x,r_2} which are the "purported" values of $F_{k^i_{x,r_1}}(g,r_1,r_2)$ and $F_{k^i_{y,r_2}}(g,r_1,r_2)$ provided by the malicious client. For the particular garbled gate entry r_1,r_2 that is being decrypted in the evaluation, we compute $\Delta_{r_1,r_2} = \bigoplus_{i\in M}(F_{k^i_{x,r_1}}(g,r_1,r_2)\oplus\delta^i_{x,r_1})\bigoplus_{i\in M}(F_{k^i_{y,r_2}}(g,r_1,r_2)\oplus\delta^i_{y,r_2})$. If Δ_{r_1,r_2} is not all zeroes in the positions containing the output and the MAC of a honest output client, then we instruct this honest output client to output $\perp$. This hybrid is statistically close to the previous hybrid from the security of MAC used in computing h.

  Hyb_3 is identically distributed to the ideal world execution.

Remark B.2. *Since the PRF used in the above construction is invoked an apriori bounded number of times, it can be replaced with a PRG that has a sufficiently large stretch.*

C Security Definitions

For self-containment, we spell out the standard security definitions for the types of protocols we construct.

C.1 Two-Sided NISC

A (malicious-secure) two-sided NISC protocol for computing a function f is given by a tuple of algorithms $(\mathsf{CRSGen}, \Pi_1, \Pi_2, \mathsf{out}_\Pi)$. CRSGen is a PPT algorithm that takes in 1^λ and outputs a crs. For each $r\in\{1,2\}$, Π_r is a PPT algorithm that takes in crs, the identity of the party (which is either R or S), the transcript seen so far, the party's private input and outputs the r-th round message in the protocol. out_Π is a deterministic algorithm that taken in the crs, the identity of the party, the full protocol transcript, the party's private input, its random tape and provides the output z.

Let $\mathcal{A}$ be a malicious adversary corrupting either the sender or the receiver in the real protocol execution. In the following, we use $\mathsf{Real}(1^\lambda,\Pi,\mathcal{A},x)$ to denote the joint distribution of the output

of the adversary and the output of the honest party (when its input is x) in the real execution of the protocol.

Let Sim be a malicious adversary that is corrupting either the sender or the receiver in the ideal protocol execution. Recall that in the ideal execution, the parties have access to a trusted functionality for computing f. The honest party sends its inputs to this trusted functionality, whereas the adversarial party can send an arbitrary value. The functionality computes the output and sends it to Sim. Sim sends the instruction to the trusted party whether to deliver this output to the honest party or abort. In the latter case, the output of the honest party is abort and in the former, the honest party outputs whatever it receives from the trusted party. We use $\mathsf{Ideal}(1^\lambda, \Pi, \mathsf{Sim}, x)$ to denote the joint distribution of the output of Sim and the output of the honest party when its input is x in the ideal execution of the protocol. Note that in this ideal execution, crs is generated by Sim.

Definition C.1. *A two-sided NISC protocol* $(\mathsf{CRSGen}, \Pi_1, \Pi_2, \mathsf{out}_\Pi)$ *is said to compute a function* f *with security against malicious adversaries if for any non-uniform PPT real world adversary* $\mathcal{A}$ *that is corrupting either the sender or the receiver, there exists a non-uniform PPT ideal world adversary* Sim *that is corrupting the same party as that of* $\mathcal{A}$ *such that for any input* x *of the uncorrupted party, we have:*

$$\mathsf{Real}(1^\lambda, \Pi, \mathcal{A}, x) \approx_c \mathsf{Ideal}(1^\lambda, \Pi, \mathsf{Sim}, x)$$

C.2 Three-Round MPC

A three-round malicious-secure n-party protocol for computing a function f is given by a tuple of algorithms $(\mathsf{CRSGen}, \Pi_1, \Pi_2, \Pi_2, \mathsf{out}_\Pi)$. CRSGen is a PPT algorithm that takes in 1^λ and outputs a crs. For each $r \in \{1, 2, 3\}$, Π_r is a PPT algorithm takes in crs, the identity of the party (which is an index from $[n]$), the transcript seen so far, the party's private input and outputs the r-th round message in the protocol. out_Π is a deterministic algorithm that takes in the crs, the identity of the party, the full protocol transcript, the party's private input, its random tape and provides the output z.

Let $\mathcal{A}$ be a malicious adversary that is corrupting an arbitrary subset M of the parties and let H be the set of honest parties. In the following, we use $\mathsf{Real}(1^\lambda, \Pi, \mathcal{A}, \{x_i\}_{i \in H})$ to denote the joint distribution of the output of the adversary and the output of the honest parties (when using input x_i for each $i \in H$) in the real execution of the protocol.

Let Sim be a malicious adversary that is corrupting the subset M of the parties in the ideal protocol execution. Recall that in the ideal execution, the parties have access to a trusted functionality for computing f. The honest parties send their inputs to this trusted functionality, whereas the adversarial party can send an arbitrary value. The functionality computes the output and sends it to Sim. Sim sends the instruction to the trusted party whether to deliver this output to the honest parties or abort. In the latter case, the output of each of the honest parties is abort and in the former, the honest parties output whatever it received from the trusted party. We denote $\mathsf{Ideal}(1^\lambda, \Pi, \mathsf{Sim}, \{x_i\}_{i \in H})$ to denote the joint distribution of the output of Sim and the output of the honest parties when their inputs are $\{x_i\}_{i \in H}$ in the ideal execution of the protocol.

Definition C.2. *A three-round* n*-party protocol* $(\mathsf{CRSGen}, \Pi_1, \Pi_2, \Pi_3, \mathsf{out}_\Pi)$ *for computing a function* f *is said to achieve security with unanimous abort against malicious adversaries if for any non-uniform PPT real world adversary* $\mathcal{A}$ *that is corrupting an arbitrary subset* M *of the parties,*

there exists a non-uniform PPT ideal world adversary Sim *corrupting* M *such that for any inputs* $\{x_i\}_{i\in[n]\setminus M}$ *of the uncorrupted parties, we have:*

$$\mathsf{Real}(1^\lambda, \Pi, \mathcal{A}, \{x_i\}_{i\in[n]\setminus M}) \approx_c \mathsf{Ideal}(1^\lambda, \Pi, \mathsf{Sim}, \{x_i\}_{i\in[n]\setminus M})$$