Round-Optimal Black-Box Protocol Compilers*

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

July 7, 2022

Abstract

We give black-box, round-optimal protocol compilers from semi-honest security to malicious security in the Random Oracle Model (ROM) and in the 1-out-of-2 oblivious transfer (OT) correlations model. We use our compilers to obtain the following black-box constructions of general-purpose protocols for secure computation tolerating static, malicious corruptions of all-but-one participants:

- A two-round, two-party protocol in the random oracle model, making black-box use of a two-round semi-honest secure protocol. Prior to our work, such a result was not known even for special functionalities such as OT. As an application, we get efficient constructions of two-round malicious OT/OLE in the random oracle model based on a black-box use of two-round semi-honest OT/OLE.
- A three-round multiparty protocol in the random oracle model, making a black-box use of two-round semi-honest OT. This protocol matches a known round complexity lower bound due to Applebaum et al. (ITCS 2020) and is based on a minimal cryptographic primitive.
- A two-round multiparty protocol in the OT correlations model, making a black-box use of a semi-malicious protocol. This improves over a similar protocol of the authors (Crypto 2021) by eliminating an adaptive security requirement and replacing nonstandard multiparty OT correlations by standard ones. As an application, we get 2-round protocols for arithmetic branching programs that make a black-box use of the underlying field.

As a contribution of independent interest, we provide a new variant of the IPS compiler (Ishai, Prabhakaran and Sahai, Crypto 2008) in the two-round setting, where we relax requirements on the IPS "inner protocol" by strengthening the "outer protocol".

1 Introduction

Minimizing the round complexity of cryptographic protocols in the presence of malicious parties has been a major theme of research in recent years. While most feasibility questions have been answered, there are still big efficiency gaps between known round-optimal protocols and their best counterparts with security against semi-honest parties.

This line of research produced many innovative ideas for bridging the efficiency gap in special cases of interest. For instance, Peikert et al. [PVW08] proposed concretely efficient 2-round

*This is a full version of [IKSS22].

†Technion.
‡UIUC.
§UCLA.
¶Tata Institute of Fundamental Research.

oblivious transfer (OT) protocols under several standard assumptions. Other concretely efficient 2-round OT protocols were proposed in [MR19, MRR20]. Chase et al. [CDI+19] and Branco et al. [BDM22] designed such protocols for oblivious linear evaluation (OLE), a natural arithmetic extension of OT. Recent techniques improve the efficiency of 2-round protocols in the batch setting, where multiple instances of OT or OLE are generated together [BCG+19b, BCG+19a]. In all these cases, efficiently obtaining security against malicious parties (without resorting to general-purpose NIZK) requires ingenious ideas that are carefully tailored to the structure of the underlying primitives. In some cases, this requires using more aggressive (and sometimes nonstandard) flavors of the assumptions that underlie the semi-honest protocols. For instance, Boyle et al. [BCG+19a] present a communication-efficient 2-round "batch-OT" protocol, realizing polynomially many instances of OT, with semi-honest security based on the Learning Parity with Noise (LPN) assumption. In the case of malicious security, they present a similar protocol in the random oracle model, but require a stronger leakage-resilient variant of LPN.

The goal of this work is to propose new general techniques for bridging the "semi-honest vs. malicious" gap (1) without increasing round complexity, (2) without strengthening the underlying assumptions, and (3) *without significantly hurting concrete efficiency.* A clean theoretical model for capturing the last requirement is a *black-box construction.* Such a construction builds a malicious-secure protocol by using an underlying semi-honest protocol *as an oracle.* This restriction ensures that the efficiency gap does not depend on the complexity or structure of the semi-honest protocol. This paradigm has been successfully applied not only in the context of theoretical feasibility results, but also in the context of concretely efficient protocols. Indeed, black-box constructions can typically be optimized to have a very low overhead, at least in an amortized sense.

There is a large body of research on such black-box constructions, including a black-box construction of constant-round *honest-majority* secure computation from one-way functions [DI05] (replacing an earlier non-black-box construction from [BMR90]), a black-box construction of malicious-secure OT from semi-honest OT [HIK+11] or trapdoor permutations [ORS15] (replacing a non-black-box construction of [GMW87]), and a black-box construction for OT extension [IKNP03] (replacing the earlier non-black-box protocol [Bea96]).

One major shortcoming of most previous black-box constructions is that they inherently increase the round complexity. In particular, they cannot be used to obtain 2-round protocols. Thus, the main question we ask is:

Can we construct round-optimal black-box transformations from semi-honest secure protocols to malicious secure variants?

The recent work of [IKSS21], building upon the IPS compiler of [IPS08], made partial progress towards settling the question. In particular, it gave a round-preserving black-box compiler that relies on a random OT correlation setup in the 2-party case, or a more complex correlated OT setup in the multiparty case. Two significant caveats are that the underlying semi-honest protocol should satisfy: *(i)* semi-malicious security;[1] and *(ii) adaptive security with erasures*, a limitation inherited from [IPS08]. This latter property is typically easy to achieve by increasing round complexity. However, it poses a major challenge in the 2-round setting. While natural two-round protocols *in*

[1]Semi-malicious security is a strengthening of semi-honest security where the adversary is allowed to choose the random tape of the corrupted parties in an arbitrary manner before the protocol begins. In the context of 2-round protocols, most (but not all) natural semi-honest protocols also satisfy this stronger security property.

the OT-hybrid model already satisfy the adaptive security requirement, standard 2-round protocols in the plain model, including semi-honest OLE or batch-OT protocols, do not.

The above state of affairs raises the following natural questions: Can we eliminate the adaptive security requirement? Can we eliminate the setup completely, or replace it by a standard OT setup in the multiparty case?

Since we are targeting 2-round protocols with security against malicious adversaries, we cannot hope to obtain results in the plain model. But since the aim of achieving black-box protocols is efficiency, this raises the natural question: can we build such round-preserving black-box protocol compilers in the *random oracle model*?

2 Our Results

In this work, we tackle both kinds questions: eliminating the adaptive security requirement and eliminating the need for correlated randomness completely in the random oracle model. In the multiparty case, we also address the goal of replacing the complex correlation setup from [IKSS21] by standard OT correlations.

In the following, when referring to 2-round *two-party* protocols, we distinguish between two types of protocols. A non-interactive secure computation (NISC) protocol [IKO+11] is a protocol for "sender-receiver" functionalities where both parties have an input but only the receiver obtains an output. Such a NISC protocol consists of a message from the receiver to the sender, followed by a message from the sender to the receiver. A *two-sided NISC* protocol is a 2-round secure protocol for general two-party functionalities, where both parties obtain an output. In such a protocol, each party sends a message in each of the two rounds, similarly to the general multiparty case.[2]

We now give a more detailed account of our results.

2.1 Round-Preserving Compilers in the OT Correlations Model

Assuming a random oblivious transfer (OT) correlations setup, we obtain the following results.

Informal Theorem 1 (Two-party protocols with OT correlations setup). *There exists a black-box compiler from any two-round semi-malicious (standard or two-sided) NISC protocol to a two-round malicious (standard or two-sided) NISC protocol given a setup that consists of random* 1*-out-of-*2 *OT correlations (alternatively, Rabin-OT correlations) between the two parties.*

See Theorem 6.6 for a formal statement in the standard NISC setting and see Section 6.3 for extension to the two-sided case.

As in the case of the IPS compiler [IPS08], the functionality f' realized by the semi-malicious protocol may depend on the target functionality f we want the malicious protocol to realize. From a feasibility point of view, it suffices to consider a semi-malicious protocol for OT (which can be used in parallel to realize f' via Yao's protocol [Yao86]). But when f is a "simple" functionality such as batch-OT[3] or batch-OLE, we can in fact use f' that consists of only a *constant* number of instances of f.

[2]While general two-sided NISC trivially implies standard NISC, the converse direction is more challenging. In particular, running two NISC instances in parallel does not yield a two-sided NISC, since a malicious party may use different inputs in the two instances.

[3]Batch-OT is not trivialized in the OT correlations model because the number of OTs in the OT correlations setup is a fixed polynomial in the security parameter.

We note that the required OT setup in the above theorem is minimal in the sense that both the number of random OT correlations and their size only depend on the security parameter and not on the circuit being computed. Moreover, recent techniques for efficient "silent" OT extension [BCG+19b] can make the setup reusable without additional interaction.

As a corollary of Informal Theorem 1, we can show that:

Informal Corollary 1 (Simple functionalities with OT-correlations setup)**.** *Given a setup consisting of fixed polynomial (in λ) number of random OT correlations, there exist a two-round batch OT/OLE protocol (respectively) with malicious security that makes black-box use of a two-round OT/OLE protocol (respectively) with semi-malicious security. Further, for the case of OLE, if the semi-malicious protocol makes black-box use of the underlying field, then so does the malicious protocol. Finally, the construction can be implemented with constant rate, namely with amortized communication cost of $O(1)$ instances of semi-malicious OT/OLE per instance of malicious OT/OLE.*

Theorem 1 improves over a similar result from [IKSS21] in that the semi-malicious protocol is not required to be adaptively secure. This enables the use of standard 2-round OLE protocols based on additively homomorphic encryption, which do not satisfy the adaptive security requirement. As an application, we get (in the OT correlations model) 2-round protocols for arithmetic formulas or branching programs over a field that make a black-box use of any 2-round semi-malicious OLE protocol over the same field. The latter, in turn, can be based (in a black-box way) on additively homomorphic encryption. This is contrasted with a generic non-arithmetic approach (e.g., via a black-box protocol for Boolean circuits [IKO+11]) that makes a non-black-box use of the field.

Theorem 1 is based on a new version of the black-box protocol compiler of [IPS08], where we replace the outer protocol with one that can be simpler and more efficient than the state-of-the-art [IKP10] protocol previously used in this setting. Besides eliminating the need for adaptive security from the semi-malicious MPC protocol, the improved outer protocol may be of independent interest.

The Multiparty Setting. In the multiparty setting, we show how to remove the complex multiparty watchlist correlations setup from the work of [IKSS21] and replace it with a simple 1-out-of-2 random OT correlations setup. Specifically,

Informal Theorem 2 (Multiparty protocols with OT-correlations setup)**.** *There exists a black-box compiler from any two-round multiparty protocol with semi-malicious security to a two-round multiparty protocol with malicious security given a setup that consists of random* 1*-out-of-*2 *OT correlations (alternatively, Rabin-OT correlations) between each ordered pair of parties.*

The formal statement appears in Theorem 7.7. As a corollary, building on [LLW20], this gives the first construction of a statistically secure 2-round protocol, with malicious security, for computing arithmetic branching programs while making a black-box use of the underlying field. The protocol relies on both an OT and OLE correlations setup.

2.2 Round-Preserving Compilers in the Random Oracle Model

Our primary contribution, which builds on the techniques developed above, is the construction of round-optimal compilers in the random oracle model.

The semi-malicious to malicious protocol compilers, described above, rely on OT correlations to perform cut-and-choose (using the watchlists mechanism introduced in [IPS08]). Our key contribution in this work is to remove the need for watchlists/OT correlations, and to instead give a novel adaptation of the *Fiat-Shamir* paradigm in the random oracle model to function as a watchlist. This gives rise to new round-optimal malicious secure protocols in the random oracle model from black-box use of semi-honest secure protocols.[4]

The Two-Party Setting. We obtain the following results in the two-party setting in the random oracle model.

Informal Theorem 3 (Two-party protocols in the ROM)**.** *There exists a black-box compiler from any (standard or two-sided) NISC protocol with semi-honest security to a (standard or two-sided) NISC protocol with malicious security in the random oracle model.*

As before, the functionality computed by the semi-honest protocol may depend on the target functionality computed by the malicious protocol. The formal statement of the transformation in the random oracle model can be found in Theorem 6.1 and its extension to the two-sided setting appears in Section 6.3.

We note that [MR17] also used the Fiat-Shamir transform to collapse the number of rounds of a sender-receiver protocol but their final protocol was not two-round and their assumptions were stronger than semi-honest two-round, two-party computation (specifically, they needed homomorphic commitments and two-round malicious secure OT protocol). Finally, NISC with semi-honest security can be obtained based on the black-box use of any two-round semi-honest oblivious transfer (OT) protocol, by relying on Yao's garbled circuits [Yao86]. This implies the following corollaries of Informal Theorem 3:

Informal Corollary 2 (Two-round OT in the ROM)**.** *There exists a construction of two-round OT with malicious security in the random oracle model that makes black-box use of two-round OT with semi-honest security.*

Informal Corollary 3 (Simple functionalities in the ROM)**.** *There exists a construction of two-round OT/OLE respectively with malicious security in the random oracle model that makes black-box use of of two-round OT/OLE respectively with semi-honest security. Further, for the case of OLE, if the semi-honest protocol makes black-box use of the underlying field, then so does the malicious protocol. Finally, the construction can be implemented with constant rate, namely with amortized communication cost of* $O(1)$ *instances of semi-honest OT/OLE per instance of malicious OT/OLE.*

Prior to our work, the only known constructions of two-round malicious OLE either made use of generic non-interactive zero knowledge, or relied on specific assumptions such as N^{th} residuosity [CDI+19] or LWE [BDM22]. The black-box constructions of two-round malicious OT required assumptions stronger than semi-honest security in the random oracle model [MR19, MRR20], or in the plain model [FMV19] (such as strongly uniform key agreement).

The Multiparty Setting. In the multiparty setting, we give a construction of a three round protocol in the random oracle model that makes black-box use of a minimal cryptographic primitive, namely a two-round semi-honest OT protocol.

[4] In the random oracle model, we additionally remove the need for semi-malicious security.

Informal Theorem 4 (Multiparty protocols in the ROM)**.** *There exists a construction of three-round MPC with malicious security in the random oracle model that makes black-box use of two-round OT with semi-honest security.*

The formal statement can be found in Theorem 7.1. Applebaum et al. [ABG$^+$20] showed that even considering only semi-honest security such a protocol is round-optimal (in the random oracle model). A recent work of Patra and Srinivasan [PS21] gave a construction of a three-round malicious secure protocol in the CRS model from any two-round malicious OT protocol in the CRS model that satisfied a certain form of adaptive security on the receiver side. In this work, we construct a black-box malicious secure protocol (in the random oracle model) by relying only on a two-round semi-honest OT.

3 Technical Overview

In this section, we describe the key ideas and techniques used in the construction of our protocol compilers.

3.1 IPS Compiler

The starting point of our work is the black-box compiler given by Ishai, Prabhakaran, and Sahai [IPS08] (henceforth, referred to as the IPS compiler). This compiler transforms a semi-honest secure protocol (with certain special properties) into a malicious secure protocol. The (simplified version of the) IPS compiler for computing a function f in the two-party setting consists of the following components:

- A client-server MPC protocol for computing f that is secure against any malicious adversary corrupting an arbitrary subset of the clients and a constant fraction of the servers. Such a protocol, requiring only two rounds, was constructed by Ishai, Kushilevitz, and Paskin [IKP10] (see also [Pas12]) making black-box use of a PRG. This protocol is referred to as the *outer protocol*.
- A semi-honest secure[5] protocol where the functionality computed by this protocol is the computation done by the servers in the outer protocol. This is referred to as the *inner protocol*.

In the IPS compiler, each party takes the role of a client in the outer MPC protocol and generates the first-round messages to be sent to the servers. The computation performed by the servers in the outer protocol is emulated by the inner protocol. Specifically, we run m instances of the inner protocol (where m is the number of servers) in parallel. In the i-th instance, the parties use as input the messages to be sent to the i-th server and use the inner protocol to compute the functionality of the i-th server. At the end of this emulation, the parties can obtain the second-round message generated by each server from the inner protocol and finally, compute the output of f using the output decoder of the outer protocol.

If the adversary cheats in an instance of the inner protocol, then this cheating translates to a corruption of the corresponding server in the outer protocol. In general, a malicious adversary may cheat in all the inner protocol instances, thereby breaking the security of each one of the virtual servers. However, note that the outer protocol is only guaranteed to be secure as long as a constant

[5]The IPS compiler required this semi-honest protocol to satisfy a variant of adaptive security with erasures. We will come back to this point soon.

fraction of the servers are corrupted. Thus, our compiler must ensure that any adversary that cheats in too many inner protocol instances gets caught. To ensure this, the IPS compiler uses a special "cut-and-choose" mechanism referred to as *watchlists*.

The simplest version of the watchlist mechanism involves a Rabin-OT channel with a carefully chosen erasure probability. For each of the m executions of the inner protocol, each party sends its input, randomness pair used in that particular execution to the other party via the Rabin OT channel. The other party then checks if the input, randomness pair for the executions it received via the channel is consistent with the transcript seen so far and aborts the execution if it detects any inconsistency. The erasure probability of the Rabin-OT channel is chosen in such a way that:

- The adversary cannot learn the private inputs of the honest parties from the information it receives via the Rabin-OT channel.
- If the adversary cheats in more than a constant fraction of the inner protocol instances, then with overwhelming probability this cheating is detected via an inconsistency by the honest party.

Thus, the watchlist mechanism ensures that a malicious adversary that cheats in more than a constant fraction of the inner protocol executions is caught and this allows us to argue the security of the compiled protocol against malicious adversaries.

Need for Adaptive Security of the Inner Protocol. As mentioned earlier, in the IPS compiler, it is not sufficient for the inner protocol to satisfy standard semi-honest security. We actually need the inner protocol to satisfy so-called "semi-malicious" security with a certain variant of adaptive security with erasures. As already noted in [IPS08], it is possible to replace semi-malicious security with standard semi-honest security using additional rounds. However, the need for adaptive security with erasure seems somewhat inherent in the proof of security. In the two-round setting, which is the primary focus of this work, this security requirement translates to a natural property of the receiver called *equivocal receiver security* [GS18]. Specifically, we require the existence of an equivocal simulator that can equivocate the first-round message of the receiver to any input. Before proceeding further, let us give some more details on why this equivocality property is needed in the security proof.

Consider an adversary that corrupts the sender and cheats in a small number of inner protocol instances. The number of such cheating executions is small enough so that it goes undetected by the watchlist mechanism. At the point of generating the first-round message from the receiver, we do not know in which executions the adversary is planning to cheat, as the receiver sends its message before the sender. Only after receiving the message from the adversarial sender, we realize that in some executions the adversary has cheated, thereby breaking the security of the inner protocol. Hence, we need to equivocate the first-round receiver message in these cheating executions to the actual receiver input so that we can derive the same output that an honest receiver obtains.

We note that this property could be added generically to certain types of protocols such as two-round semi-honest oblivious transfer. However, it is not known how to add this property to general protocols by making black-box use of cryptography. Even for special cases such as Oblivious Linear Evaluation (OLE), we do not know of any method to add this property to natural semi-honest OLE instantiations.

3.2 A New Compiler: Removing Equivocality

In this work, we give a new IPS-style compiler in the two-round setting where the inner protocol need not satisfy the equivocal receiver message property.

Strengthening the Outer Protocol. Our main idea to achieve this is to strengthen the requirements from the outer MPC protocol. Namely, we show that if the outer protocol satisfies a certain output error-correction property, then we do not need equivocal receiver security from the inner protocol. Our output error-correction property requires that for all choices of second-round messages from the (few) corrupted servers, the output of the honest receiver remains the same. Indeed, we can substitute the outputs of those cheating executions with any default value and still we are guaranteed to obtain the same output as that of an honest receiver. This removes the need to equivocate the first-round message of the receiver for the executions where the adversary is cheating and instead, we can rely on any semi-malicious inner protocol. The main question we are now tasked with solving is to construct an outer protocol in the client-server setting that runs in two rounds and satisfies the output error-correction property.

Barriers. We first observe that if the outer protocol satisfies guaranteed output delivery, then it satisfies the error correction property as well. Unfortunately, Gennaro et al. [GIKR02] showed that in the two-round setting, if more than one party is corrupted, then it is impossible to construct protocols that have guaranteed output delivery. Indeed, we do not know of any ways to bypass this impossibility result even to achieve the weaker goal of error correction.

Pairwise Verifiable Adversaries. To overcome this barrier, we show that it is sufficient to achieve error correction against a restricted class of adversaries, that we call *pairwise verifiable*. In this model, the adversary that is corrupting either one of the two clients and a constant fraction of the servers is forced to send a first-round message from the corrupted client to the honest servers such that these messages pass a specified pairwise predicate check. Namely, there is a predicate that takes the first-round messages sent to any two servers and outputs either accept or reject. We require the first-round messages sent by the adversary to each pair of honest servers to pass this predicate check. However, the first-round messages sent between corrupted servers or between an honest server and a corrupted server need not satisfy the pairwise verification check. Additionally, second-round messages from corrupted servers can be generated arbitrarily. We show that once we restrict the adversary to be pairwise verifiable, we can construct simple and efficient outer protocols that also satisfy output error correction. In particular, we show that the semi-honest secure protocol from [IK00] is secure against pairwise verifiable adversaries if we replace the plain Shamir secret sharing with a bi-variate Shamir secret sharing [BGW88]. The error correction property of this construction can be shown by viewing Shamir secret sharing as an instance of the Reed-Solomon error correcting codes.

Why is security against Pairwise Verifiable Adversaries sufficient? We now explain why this weaker security notion is sufficient to instantiate the IPS compiler for two-round protocols. To see why this is the case, we modify the watchlist mechanism so that it not only checks if the pair of input and randomness it received via the Rabin-OT channel is consistent with the transcript, but also checks if the inputs (a.k.a. the first-round messages sent to the servers) pass the pairwise verification check. Using standard statistical arguments, we show that if all the inputs received via the Rabin-OT channel pass the pairwise verification check, then a large fraction of the other messages also pass the pairwise verification checks. This translates to the adversary only corrupting a small fraction of the servers and we can rely on the security of the outer protocol against pairwise verifiable adversaries.

Here, we point out that while restricting the adversary to be pairwise verifiable is sufficient for our purposes, our techniques suggest that security requirements from the outer protocol could potentially be further weakened to (say) only require security against adversaries that pass more general multi-server consistency checks, for example, those that are three-wise (as opposed to pairwise) verifiable.

Instantiating the Rabin-OT Channel. We now explain how to instantiate a Rabin-OT channel if we have access to 1-out-of-2 OT correlations:

1. We first transform the 1-out-2 OT correlations non-interactively to 1-out-of-p correlations. Such a transformation is implicit in the work of [BCR86].

2. We then use the transformation described in [IPS08, Section 2] to convert 1-out-of-p random OT correlations into a single-round Rabin OT protocol with erasure probability $1 - 1/p$.

We show that such a rational erasure probability is sufficient to instantiate the IPS compiler.

3.3 Protocol Compiler in the Random Oracle Model

To give a compiler in the random oracle model, we first observe that the Rabin OT channel can be replaced with a k-out-of-m OT channel (for an appropriate choice of k) and the same arguments go through. Our key idea here is to replace the k-out-of-m OT channel with the Fiat-Shamir transformation [FS87] applied using a random oracle. Specifically, we require both parties to additionally send a non-interactive and extractable commitment to their input and randomness used in each of the inner protocol instances.[6] In each round, we require the party sending the message to hash the transcript seen so far along with the messages generated in this round to obtain a set of executions (called the opened executions) of size k. The party, in addition to sending the messages of the inner protocol instances in that particular round, must also reveal the input–randomness pair (via an opening of the commitments) for the opened executions. The other party checks that the openings are correct, the random oracle output is correctly computed, the input–randomness pair in the opened executions are consistent with the transcript seen so far, and all pairwise consistency checks pass.

In the security proof, we rely on the correlation-intractability of the random oracle [CGH04] to show that if the adversary cheats in more than a constant fraction of the inner protocol instances, then with overwhelming probability the opened executions will intersect with the cheating executions. This will therefore be detected by the honest party forcing it to abort. In our proof of security, we also rely on the programmability of the random oracle to pre-determine the set of opened executions of the honest parties.

[6]Such a commitment can be constructed unconditionally in the random oracle model [Pas03].

Relying on a Semi-Honest Secure Protocol. We observe that in the random oracle model, it is sufficient for the inner protocol to satisfy semi-honest security rather than semi-malicious security. Specifically, the random tape used by each party in an instance of the inner protocol is set to be the output of the random oracle on the party index, the instance number, and a randomly chosen salt. This ensures that even if the salt is not uniformly random, the adversarial parties will query the random oracle on different inputs which implies that the outputs obtained from the oracle will be uniform and uncorrelated.

3.4 Two-Sided NISC

In the protocol compiler described earlier, at the end of the second round, the receiver obtains the output of the two-party functionality whereas the sender does not obtain any output. To extend this protocol to the setting where both parties get the output (called the two-sided NISC setting [IKSS21]), we cannot use the naïve idea of running the one-sided protocol in parallel but in opposite directions. Specifically, nothing prevents a cheating adversary from using inconsistent inputs in both these executions, thereby, breaking the security of the overall protocol. To prevent this attack, we further refine the IPS compiler methodology. We modify the first round commitments/message sent via the Rabin-OT channel to include the inputs and the randomness used on both sides of the inner protocols. In the opened/non-erased executions, in addition to the checks that are already performed, each party checks if the inputs used on both sides are the same and if it is not the case, then the honest parties abort. This prevents the adversary from using inconsistent inputs in "many" instances of the inner protocol, and if that is the case, we can rely on the security of the outer protocol to show that this adversary does not learn any additional information about the honest party inputs.

3.5 The Multiparty Setting

In extending the above ideas to the multiparty setting, we face two main challenges:

1. First, we do not know of any two-round black-box inner protocol in the semi-honest setting (and indeed [ABG+20] gave some barriers). Moreover, in existing three-round protocols [PS21], if the adversary cheats in generating the first-round message, then the adversary can recover the private inputs of the honest parties. Thus, we need the first message in the (3-round) inner protocol to satisfy a certain form of adaptive security with erasures even if the outer protocol has the output error correction property.

2. Recall that to use the security of the semi-honest inner protocol, we need to additionally give the simulator the power to program the random tape of the corrupted parties in some intermediate hybrids. Note that in our compiler we rely on the random oracle to perform this programming. However, a cheating adversary on behalf of a corrupted party i could query the random oracle on many different salts where the first two parts of the query are fixed to the same i and instance number j. It could then use the output of any one of these queries as the random tape in the j-th inner protocol instance. A natural idea to deal with this is to choose one of these queries uniformly at random and "embed" the programmed random tape as the output of the chosen query. The hope is that the adversary chooses this particular query with non-negligible probability and we can use this to come up with a reduction that breaks the security of the inner protocol. But this idea quickly runs into trouble in the multiparty

setting as the adversary could potentially corrupt an arbitrary subset of the parties, and we require the adversary on behalf of each malicious party to correctly choose this embedded query. This only happens with probability that is exponential in n (where n is the number of parties) and is not sufficient to break the security of the inner protocol.

To solve the first issue, we show how to add the required equivocal properties to the protocol of [PS21] in a black-box manner relying only on two-round semi-honest OT. This allows us to use it as the inner protocol and instantiate the IPS compiler.

To solve the second issue, we rely on the fact that the semi-honest secure protocol in [PS21] has a special structure. Namely, it is a parallel composition of a sub-protocol that computes a special functionality called $\mathsf{3MULTPlus}$. Importantly, for this discussion it is sufficient to note that $\mathsf{3MULTPlus}$ is a three-party functionality. The security of the composed protocol is argued via a hybrid argument where we switch each one of these sub-protocols for computing the $\mathsf{3MULTPlus}$ functionality to the ideal world. Now, relying on this special structure, we show that in the intermediate hybrids, it is sufficient to program the random tapes of the corrupted parties that participate in a single instance of the sub-protocol. Since the number of such parties is only a constant, we can show that the adversary chooses the "correct" random oracle outputs with non-negligible probability and this allows us to provide a reduction that breaks the security of the sub-protocol.

4 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.

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 $|\Pr[D(1^\lambda, X_\lambda) = 1]| - \Pr[D(1^\lambda, Y_\lambda) = 1]| \leq \mathsf{negl}(\lambda)$.

4.1 Semi-Honest Two-Round Two-Party Computation

We now give the syntax and definition for a two-round semi-honest two-party computation protocol.

Syntax. Consider two parties, a sender with input y and a receiver with input x. Let f be an arbitrary two-party functionality. A two-party protocol Π for computing f is given by a tuple of algorithms $(\Pi_1, \Pi_2, \mathsf{out}_\Pi)$. Π_1 is run by the receiver and takes as input 1^λ and the receiver input x and outputs (π_1, sk). The receiver sends π_1 to the sender in the first round. Π_2 is run by the sender and it takes as input 1^λ, π_1, and the sender input y and outputs π_2. The sender sends π_2 to the receiver in the second round. The receiver then runs out_Π on inputs π_2 and sk and obtains the output z. Let $\mathsf{View}_R(\langle R(1^\lambda, x), S(1^\lambda, y)\rangle)$ and $\mathsf{View}_S(\langle R(1^\lambda, x), S(1^\lambda, y)\rangle)$ be the views of the receiver and the sender during the protocol interaction with inputs x and y respectively. Here, View of a party (either the sender or the receiver) includes its private input, its random tape, and the transcript of the protocol. The protocol Π satisfies the definition given below.

Definition 4.1 (Semi-Honest Security). *A two-round, two-party protocol* $\Pi = (\Pi_1, \Pi_2, \mathsf{out}_\Pi)$ *is said to securely compute* f *against semi-honest adversaries if it satisfies the following properties:*

- ***Correctness:*** *For every receiver's input x and for every sender input y, we have:*

$$\Pr[\mathsf{out}_\Pi(\pi_2, sk) = f(x,y)] = 1$$

where $(\pi_1, sk) \leftarrow \Pi_1(1^\lambda, x)$ and $\pi_2 \leftarrow \Pi_2(1^\lambda, \pi_1, y)$.

- ***Security:*** *There exists a simulator Sim_Π such that for any receiver's input x and sender's input y, we have:*

$$\mathsf{View}_S(\langle R(1^\lambda, x), S(1^\lambda, y)\rangle) \approx_c (y, r, \mathsf{Sim}_\Pi(1^\lambda, R, y))$$

$$\mathsf{View}_R(\langle R(1^\lambda, x), S(1^\lambda, y)\rangle) \approx_c (x, r, \mathsf{Sim}_\Pi(1^\lambda, S, (x,r), f(x,y)))$$

where the random tape r of the sender/receiver in the second distribution is uniformly chosen.

Remark 4.2. *In the standard definition of semi-honest security, Sim_Π is allowed to additionally set the random tape of the corrupted receiver. Here, we consider a slightly stronger definition where the random tape of the corrupted receiver is chosen uniformly and this is provided as input to Sim_Π and Sim_Π is required to produce the transcript of the protocol. We note that this definition is implied by the standard definition whenever f is reverse sampleable. Specifically, given $(x, f(x,y))$, if there is an efficient algorithm I that outputs some y' s.t. $f(x,y) = f(x',y')$ then the weaker definition implies the stronger definition described above. Indeed, for most natural functionalities, such as Oblivious Transfer (OT), Oblivious Linear Evaluation (OLE), their batched versions, batch-OT and batch-OLE, there exists such a reverse sampler, and the above definition is satisfied by all semi-honest secure protocols.*

4.2 Semi-Malicious Two-Round Two-Party Computation

Semi-Malicious security [AJL+12] is a strengthening of the semi-honest security definition where we additionally allow the adversary to choose the random tape of the corrupted party arbitrarily. However, the adversary is restricted to follow the protocol specification. Such an adversary is called as a semi-malicious adversary. A two-round semi-malicious secure two-party protocol has the same syntax of a semi-honest protocol and satisfies the definition given below.

Definition 4.3 (Semi-Malicious Security)**.** *A two-round, two-party protocol $\Pi = (\Pi_1, \Pi_2, \mathsf{out}_\Pi)$ is said to securely compute f against semi-malicious adversaries if it satisfies the following properties:*

- ***Correctness:*** *For every receiver's input x and for every sender input y, we have:*

$$\Pr[\mathsf{out}_\Pi(\pi_2, sk) = f(x,y)] = 1$$

where $(\pi_1, sk) \leftarrow \Pi_1(1^\lambda, x)$ and $\pi_2 \leftarrow \Pi_2(1^\lambda, \pi_1, y)$.

- ***Security:*** *There exists a simulator Sim_Π such that for any receiver's input x, sender's input y and for any receiver's random tape r, we have:*

$$\mathsf{View}_S(\langle R(1^\lambda, x), S(1^\lambda, y)\rangle) \approx_c \mathsf{View}_S(\langle R(1^\lambda, \mathbf{0}), S(1^\lambda, y)\rangle)$$

$$\mathsf{View}_R(\langle R(1^\lambda, x, r), S(1^\lambda, y)\rangle) \approx_c (x, r, \mathsf{Sim}_\Pi(1^\lambda, S, (x,r), f(x,y)))$$

where $\mathbf{0}$ is a default input.

4.3 Extractable Commitments in ROM

In our protocol compilers, we make use of non-interactive, straight-line extractable commitments in the random oracle model. Namely, the commitments are computationally hiding and straight-line extractable by observing the queries that the adversary makes to the random oracle. Such commitments were constructed in [Pas03].

4.4 Pairwise Verifiable Secret Sharing

Consider a linear t-out-of-m threshold secret sharing scheme where the secrets are over a finite field $\mathbb{F}$ and the shares are over another finite field $\mathbb{F}'$. We use $+$ and $\cdot$ to denote the addition and multiplication operations over both the fields.

Definition 4.4 (Pairwise Verifiable Predicate)**.** *A predicate P is a pairwise verifiable predicate if it takes a threshold t, two indices $j, k \in [m]$ and the purported j-th and k-th shares x_j and x_k and outputs 1/0. Further, if $P(t, j, k, (x_j, x_k)) = 1$ and $P(t, j, k, (x'_j, x'_k)) = 1$, then $P(t, j, k, (x_j + x'_j, x_k + x'_k)) = 1$ and $P(2t, j, k, (x_j \cdot x'_j, x_k \cdot x'_k)) = 1$.*

In the main body, we also extend the definition of the pairwise verifiable predicate P to take in a vector of pair of shares and apply the above pairwise check for each pair.

Definition 4.5 (Pairwise Verifiable and Error Correctable Secret Sharing)**.** *A t-out-of-m threshold linear secret sharing scheme $(\mathsf{Share}_{(t,m)}, \mathsf{Rec}_{(t,m)})$ is said to be k-multiplicative and ℓ-error-correctable w.r.t. pairwise predicate P if:*

1. ***k-Multiplicative**: Given m shares of elements $x_1, \ldots, x_k$ arranged as a matrix M of k rows and m columns, the row vector obtained by computing the product of each column of M is a kt-out-of-m secret sharing of $x_1 \cdot x_2 \ldots \cdot x_k$.*

2. ***Pairwise Verifiable Error Correction**: Let T be a subset of $[m]$ of size at most ℓ. Let $(x_1, \ldots, x_m)$ be arbitrary elements such that for any threshold $t' \leq kt$ and for any $j, k \in [m] \setminus T$, $P(t', j, k, x_j, x_k) = 1$. Then, for any $\{\overline{x}_i\}_{i \in T}$, $\mathsf{Rec}_{(t',m)}(\{x_i\}_{i \in T}, \{x_i\}_{i \notin T}) = \mathsf{Rec}_{(t',m)}(\{\overline{x}_i\}_{i \in T}, \{x_i\}_{i \notin T}) = x$. Furthermore, there exists an efficient procedure* Extrapolate *that on input t', $\{x_i\}_{i \notin T}$ outputs $\{x'_i\}_{i \in T}$ such that $(\{x_i\}_{i \notin T}, \{x'_i\}_{i \in T}) \in \mathsf{supp}(\mathsf{Share}_{(t',m)}(x))$.*

The above definition of pairwise verifiable secret sharing is the same as the one given in [IKP10] except that we additionally need error correction property as well. We note that bivariate Shamir secret sharing is a t-out-of-m secret sharing scheme that is k-multiplicative and ℓ-error correctable as long as $m \geq kt + 2\ell + 1$. The pairwise predicate corresponds to equality checking of polynomial evaluations.

5 Two-Round Client-Server Protocol with Pairwise Verifiability

In this section, we give a construction of a two-round, pairwise verifiable MPC protocol in the client-server model. We start with the Definition of this protocol in Section 5.1.

5.1 Definition

Syntax. Let f be an arbitrary n-party functionality. Consider the standard client-server MPC setting [DI05] with n clients and m servers. A two-round protocol $\Phi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ for computing a function f in this model has the following syntax:

- $\mathsf{Share}(1^\lambda, i, x_i)$: It outputs a set of shares $(x_1^i, \ldots, x_m^i)$ along with a verification key vk_i.
- $\mathsf{Eval}(j, (x_j^1, \ldots, x_j^n))$: It outputs a string ϕ_j.
- $\mathsf{Dec}(i, vk_i, (\phi_1, \ldots, \phi_m))$: It outputs a string z or the special symbol $\perp$.

In the first round of the protocol, each client $i \in [n]$ runs the algorithm Share on its private input x_i and obtains a set of shares $(x_1^i, \ldots, x_m^i)$ and a verification key vk_i. It then sends x_j^i as the first round message to the j-th server for each $j \in [m]$. In the second round, each server $j \in [m]$ runs the Eval algorithm on the first round messages received from each client and obtains the string ϕ_j. A subset of the clients are designated as output clients in the protocol. The j-th server sends ϕ_j to each of the output clients in the second round. To obtain the output, each output client i runs Dec on its verification key vk_i and the second round messages received from all the servers to obtain the output z.

Security Definition. Below we provide the security definition of a client-server MPC protocol that is pairwise verifiable w.r.t. predicate P.

Definition 5.1 (Admissible Adversary)**.** *Let P be a pairwise predicate that takes a client index $i \in [n]$, two server indices $j, k \in [m]$, the first round message (x_j^i, x_k^i) sent by the i-th client to the servers j and k and outputs 1/0. An adversary $\mathcal{A}$ corrupting a subset of the clients and up to t servers is said to be admissible w.r.t. pairwise predicate P if for every honest pair of servers j, k and every corrupted client i, the output of the predicate P on input $(i, j, k, (x_j^i, x_k^i))$ is 1.*

Definition 5.2 (Pairwise Verifiable MPC)**.** *Let f be a n-party functionality. A protocol $\Phi =$ $(\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ is a two-round, n-client, m-server pairwise verifiable MPC protocol for computing f against t server corruptions if there exists a pairwise predicate P such that:*

1. ***Error Correction:*** *If $\mathcal{A}$ is any admissible adversary (see Definition 5.1) w.r.t. P corrupting a subset T (where $|T| \leq t$) of the servers and for any two sets of second round messages $\{\phi_j\}_{j \in T}$ and $\{\overline{\phi}_j\}_{j \in T}$ and for any honest client $i \in [n]$, $\mathsf{Dec}(i, vk_i, \{\phi_j\}_{j \notin T}, \{\phi_j\}_{j \in T}) =$ $\mathsf{Dec}(i, vk_i, \{\phi_j\}_{j \notin T}, \{\overline{\phi}_j\}_{j \in T})$ where $\{\phi_j\}_{j \notin T}$ are the second round messages generated by the honest servers in the interaction with $\mathcal{A}$ and vk_i is the verification key output by Share algorithm.*

2. ***Security:*** *For any admissible adversary $\mathcal{A}$ (see Definition 5.1) w.r.t. P corrupting a subset of the clients and (adaptively) corrupting upto t servers, there exists an ideal world simulator Sim_Φ such that for any choice of inputs of the honest clients, the following two distributions are computationally indistinguishable:*

   - ***Real Execution.*** *The admissible adversary $\mathcal{A}$ interacts with the honest parties who follow the protocol specification. The output of the real execution consists of the output of the admissible adversary $\mathcal{A}$ and the output of the honest output clients.*
   - ***Ideal Execution.*** *This corresponds to the ideal world interaction where Sim_Φ and the honest client have access to the trusted party implementing f. Each honest client sends its input to f and each honest output client outputs whatever the trusted functionality*

sends back. For every honest output client, Sim_Φ *sends a special instruction to the trusted functionality to either give the output of* f *to the output client or the special symbol* $\perp$*. The output of the ideal execution corresponds to the output of* Sim_Φ *and the output of all the honest outputs clients.*

5.2 Construction

In this section, we give a construction of pairwise verifiable MPC protocol based on a 4-multiplicative t-out-of-m secret sharing scheme that is t-error-correctible w.r.t. pairwise predicate P.

5.2.1 Protocol for $\mathcal{SREN}$

Recall that the complexity class $\mathcal{SREN}$ consists of the set of all functions that have a degree-3 randomized encoding [IK00, AIK04]. These include log-depth arithmetic circuits and arithmetic branching programs.

Theorem 5.3. *Let* $(\mathsf{Share}_{(t,m)}, \mathsf{Rec}_{(t,m)})$ *be a* t*-out-of-*m*, 4-multiplicative,* t*-error-correctable secret sharing secret sharing scheme w.r.t. pairwise predicate* P *(see Definition 4.5). Let* f *be an arbitrary* n*-party functionality in the complexity class* $\mathcal{SREN}$*. Then, there exists a construction of an* n*-client,* m*-server pairwise verifiable MPC protocol for computing* f *against* t *server corruptions (see Definition 5.2). The computational cost of the protocol is polynomial in* n *and the size of the branching program for computing* f*.*

Notation. We give a protocol to compute a collection of degree-3 polynomials $p_1, \ldots, p_r$ over a finite field $\mathbb{F}$ on the private inputs of the clients. The construction for any function in $\mathcal{SREN}$ follows via standard reduction using randomized encoding [IK00, AIK04].[7] Let MAC be a strongly unforgeable one-time MAC scheme where the tag generation algorithm is computing a degree-1 function over $\mathbb{F}$. For instance, $\mathsf{MAC}_{\mathbf{a},\mathbf{b}}(x) = x\mathbf{a} + \mathbf{b}$ satisfies this property. Let g be an augmented functionality that takes x_i and a collection of r one-time MAC keys denoted by $k_i = \{k_{i,j}\}_{j\in[r]}$ from the i-th client and outputs $(p_1(x_1, \ldots, x_n), \ldots, p_r(x_1, \ldots, x_n))$ along with $\{\sigma_i = \{\mathsf{MAC}_{k_{i,j}}(p_j(x_1, \ldots, x_n))\}_{j\in[r]}\}_{i\in[n]}$. Note that g is a degree-4 function in the inputs $\{(x_i, k_i)\}_{i\in[n]}$ of the clients.

Description of the Protocol.

- $\mathsf{Share}(i, x_i)$: It samples uniform one-time MAC keys $k_i = \{k_{i,j}\}_{j\in[r]}$ and generates a secret share of each element in (x_i, k_i) using $\mathsf{Share}_{(t,m)}$. In addition to this, for each output element of g, the i-th client generates the shares of 0 using $\mathsf{Share}_{(4t,m)}$. The shares sent to the j-th server are the j-th share of (x_i, k_i) and the j-th shares of 0 corresponding to each output element of g. The verification key $vk_i = k_i$.
- $\mathsf{Eval}(j, x_j^1, \ldots, x_j^n)$: For each output element of g, the j-th server computes the j-th share of a $4t$-out-of-m secret sharing of this element using the 4-multiplicative property of the underlying secret sharing scheme. It then adds the corresponding j-th secret share of 0 received from each client to this share and refreshes it. ϕ_j comprises of the refreshed j-th share for each output element of g as computed above.

[7] The complexity class $\mathcal{SREN}$ consists of the class of circuits that takes inputs in $\{0, 1\}$. To convert into functions that takes elements from a finite field $\mathbb{F}$ (of size $p = \mathsf{poly}(n)$), we take each field element a and compute $a^{p-1} \mod p$. This gives a 0/1 value and it can be computed by a branching program of length polynomial in p.

- $\mathsf{Dec}(i, vk_i, \phi_1, \ldots, \phi_m)$: To compute the output, Dec first reconstructs the output of g by running on $\mathsf{Rec}_{(4t,m)}$ on $\phi_1, \ldots, \phi_m$. It parses this as $(z, \sigma_1, \ldots, \sigma_n)$. It then uses the verification key vk_i to check if σ_i is a valid tag on the message z. If it is the case, it outputs z and otherwise, it outputs $\perp$.

Predicate P'. The predicate P' in the definition of admissible adversary corresponds to the pairwise verification of each share obtained from the client. This includes the t-out-of-m secret sharing of the private inputs (x_i, k_i) as well as each of the $4t$-out-of-m secret sharing of 0.

Error Correction. Let $\mathcal{A}$ be any admissible adversary corrupting a subset T of the servers of size at most t. This implies that each of the input shares sent by $\mathcal{A}$ to every pair of honest servers pass the pairwise verifiability check. Since the refreshed output shares in ϕ_j for each $j \in [m]$ are computed as a degree-4 polynomial, it follows from Definition 4.4 that for each pair of honest servers j, k, these refreshed shares in ϕ_j and ϕ_k pass the pairwise verifiability check. Thus, the error correction property of the protocol Φ directly follows from the pairwise verifiable error correction property of the underlying secret sharing scheme.

Security. Let $\mathcal{A}$ be an admissible adversary corrupting a subset of the clients denoted by M and upto t servers denoted by S.

1. For each of the corrupted servers, Sim_Φ sends the corresponding secret secret share of a default value to $\mathcal{A}$ on behalf of each of the honest clients.

2. Sim_Φ receives the first round messages sent to the honest servers from $\mathcal{A}$. Since these messages are guaranteed to be pairwise verifiable, Sim_Φ extracts the input $\{(x_i, k_i)\}_{i \in M}$ (using $\mathsf{Rec}_{(t,m)}$) from these messages. It sends x_i to the ideal functionality.

3. Based on the shares received from the malicious clients, Sim_Φ also computes the sum of the vector of values denoted by Δ sent by the malicious clients as the purported sharing of vector of 0 (one entry corresponding to each output element of g). If this vector is not all zeroes in the positions corresponding to the output z and the MAC of some honest output client, then Sim_Φ instructs the ideal functionality to output $\perp$ to this honest output client.

4. If some output client is corrupted, then Sim_Φ obtains the output z of the function f from the ideal functionality. It computes σ_i as described in g for each $i \in M$ and samples $\{\sigma_i\}_{i \in [n] \setminus M}$ uniformly. It computes the purported shares of the malicious client inputs sent to the corrupted servers using the shares received by the honest servers (by making use of the $\mathsf{Extrapolate}$ algorithm). Using these input shares, it computes the shares of the output computed by the corrupted servers if they follow the protocol. Conditioned on fixing these shares, it generates a uniform $4t$-out-of-m secret sharing of $(z, \sigma_1, \ldots, \sigma_n) + \Delta$ and sends the shares of the honest servers to the adversary.

5. Sim_Φ outputs whatever the adversary outputs.

We now argue that the real execution and the ideal execution are statistically close by a hybrid argument.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the output of the real execution.

- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we do the following:
  - Based on the shares sent to the honest servers, we extract $\{(x_i, k_i)\}_{i \in M}$ and also compute Δ as described in the simulation.
  - We compute the output $(z, \{\sigma_i\}_{i\in[n]}) + \Delta$.
  - We compute the purported shares of the malicious client inputs sent to the corrupted servers based on the shares received by the honest servers (using the $\mathsf{Extrapolate}$ algorithm). Using these input shares, we compute the shares of the output obtained by the corrupted servers if they followed the protocol.
  - Conditioned on the fixing the above computed output shares, we sample the second round message from the honest servers as fresh shares of $4t$-out-of-m secret sharing of $(z, \{\sigma_i\}_{i\in[n]}) + \Delta$.

  This hybrid is identical to the previous hybrid since the adversary $\mathcal{A}$ is admissible and the honest client sends a $4t$-out-of-m secret sharing of vector of zeroes.
- $\underline{\mathsf{Hyb}_2}$: In this hybrid, for every honest client, we replace the output shares from the corrupted servers to be some arbitrary values. It follows from the error correction property that Hyb_1 and Hyb_2 are identical.
- $\underline{\mathsf{Hyb}_3}$: In this hybrid, if Δ is not all zeroes string in the positions corresponding to the output z or the MAC of some honest output client, we instruct that honest output client to output $\perp$. This hybrid is statistically close to Hyb_2 from the strong unforgeabilty of the one-time MAC scheme.
- $\underline{\mathsf{Hyb}_4}$: In this hybrid, we sample $\{\sigma_i\}_{i\in[n]\setminus M}$ uniformly. This hybrid is identical to the previous hybrid from the uniformity of the MACs of a one-time MAC scheme.
- $\underline{\mathsf{Hyb}_5}$: In this hybrid, we replace the shares of the honest clients sent to the corrupted servers to be shares of default value. This hybrid is identically distributed to the previous one from the perfect privacy of the underlying secret sharing scheme. This hybrid is identical to the output of the ideal execution.

This completes the proof of the theorem.

5.2.2 Protocol for Arbitrary Circuits

Theorem 5.4. *Let $(\mathsf{Share}_{(t,m)}, \mathsf{Rec}_{(t,m)})$ be a t-out-of-m, 4-multiplicative, t-error-correctable secret sharing scheme w.r.t. pairwise predicate P (see Definition 4.5). Let f be an arbitrary n-party functionality. Then, there exists a construction of an n-client, m-server pairwise verifiable MPC protocol for computing f against t server corruptions (see Definition 5.2) that makes black-box use of a PRF. Furthermore,* Eval *algorithm does not perform any cryptographic operations. The computational cost of the protocol is polynomial in the circuit size of f, the security parameter 1^λ, and the number of parties.*

Notation. We recall the BMR garbled circuit [BMR90] construction. 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^i_w and two random PRF keys $k^i_{w,0}, k^i_{w,1} \leftarrow \{0,1\}^\lambda$. If w is the input wire of some party P_j, then $b^i_w = 0$ and $k^i_{w,0} = 0^\lambda$ and $k^i_{w,1} = 0^\lambda$ for each $i \neq j$. If w is the output wire, then $b^i_w = 0$ and $k^i_{w,0} = 0^\lambda$ and $k^i_{w,1} = 0^\lambda$ for each $i \in [n]$. We use b_w to denote $\bigoplus_{i=1}^n b^i_w$.

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:
$$\widetilde{G}_{r_1,r_2} = \Big(\bigoplus_{i=1}^{n} F_{k^i_{x,r_1}}(g,r_1,r_2) \oplus \bigoplus_{i=1}^{n} F_{k^i_{y,r_2}}(g,r_1,r_2)\Big) \oplus \Big(\{k^i_{z,\chi_{g,r_1,r_2}}\}_{i\in[n]}, \chi_{g,r_1,r_2}\Big)$$
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 $k^i_{w,x_w\oplus b_w}$ 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 $(\{k^i_{z,\chi_{g,r_1,r_2}}\}_{i\in[n]}, \chi_{g,r_1,r_2})$ in the above garbled gate gadget for each gate g and $r_1, r_2 \in \{0,1\}$ and $(k^i_{w,x_w\oplus b_w}, x_w \oplus b_w)$ for each input wire w is a vector of degree-3 polynomials in the inputs of the parties. Let g' be the sequence of all such degree-3 polynomials in the computation of the BMR garbling gadget and g be the augmented functionality (that includes the tags computed on each output of g') defined in the previous section.

Description of the Protocol. We give a protocol for computing the BMR garbled circuit.

- $\mathsf{Share}(1^\lambda, i, x_i)$: The i-th client uses the sharing procedure from Section 5.2.1 to securely evaluate g on its private inputs. In addition, i-th client evaluates $F_{k^i_{x,r_1}}(g,r_1,r_2)$ and $F_{k^i_{y,r_2}}(g,r_1,r_2)$ for each gate g, $r_1, r_2 \in \{0,1\}$ and generates a $4t$-out-of-m secret sharing of these values using $\mathsf{Share}_{(4t,m)}$. The verification key corresponds to the verification key output by the sharing procedure from the previous section.
- $\mathsf{Eval}(1^\lambda, j, x^1_j, \ldots, x^n_j)$: Each server computes the $4t$-out-of-m refreshed shares of each output of g as in the previous protocol. It then adds the shares of $F_{k^i_{x,r_1}}(g,r_1,r_2)$ and $F_{k^i_{y,r_2}}(g,r_1,r_2)$ from each $i \in [n]$ to the refreshed share of $(\{k^i_{z,\chi_{g,r_1,r_2}}\}_{i\in[n]}, \chi_{g,r_1,r_2})$ for each gate g and $r_1, r_2 \in \{0,1\}$.
- $\mathsf{Dec}(1^\lambda, i, vk_i, \phi_1, \ldots, \phi_j)$: The output client reconstructs the BMR garbled gadget from the output shares using $\mathsf{Rec}_{(4t,m)}$. It then starts evaluating the BMR garbled gadget, and for every gate once it recovers $(\{k^i_{z,\chi_{g,r_1,r_2}}\}_{i\in[n]}, \chi_{g,r_1,r_2})$ (for some r_1, r_2), Dec checks using the verification key vk_i whether the recovered value passes the verification check from the previous section. If yes, it proceeds with the evaluation and otherwise, it aborts.

Predicate P'. The predicate P' in the definition of admissible adversary corresponds to the predicate defined in the previous section and the pairwise checks for the shares of $F_{k^i_{x,r_1}}(g,r_1,r_2)$ and $F_{k^i_{y,r_2}}(g,r_1,r_2)$ sent by the client for each gate g and $r_1, r_2 \in \{0,1\}$.

Sketch of Proof of Security. The error correction property is argued in an identical fashion to the error correction in the previous section. 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.

- $\underline{\mathsf{Hyb}_0}$: 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 g from the previous section to simulate the view of $\mathcal{A}$ and generate the outputs of all the honest output clients. This hybrid

is indistinguishable to the previous one from the security of protocol for computing g described in the previous section.

- $\underline{\mathsf{Hyb}_2}$: 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.
- $\underline{\mathsf{Hyb}_3}$: In this hybrid, we extract the "purported" $F_{k^i_{x,r_1}}(g, r_1, r_2)$ and $F_{k^i_{y,r_2}}(g, r_1, r_2)$ sent by the adversary on behalf of some malicious client i. Let δ^i_{x,r_1} and δ^i_{x,r_2} be the actual values that are being secret shared. We also extract k^i_{x,r_1} and k^i_{x,r_2} using the simulator for computing g. 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 g from the previous section. Hyb_3 is identically distributed to the ideal world execution.

Remark 5.5. *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.*

6 Black-Box Protocol Compilers in the Two-Party Setting

In this section, we give our black-box protocol compilers to construct round-optimal malicious-secure protocols in the two-party setting. In Section 6.1, we give our compiler in the random oracle model. In Section 6.2, we give our compiler in the OT correlations model. Finally, in Section 6.3, we show how to extend these compilers to give a round-optimal, malicious-secure, two-party protocol in the two-sided setting.

6.1 Protocol Compiler in the Random Oracle Model

In this subsection, we give a black-box compiler that transforms from any two-round semi-honest two-party protocol to a two-round malicious secure protocol in the random oracle model. We state the formal theorem statement below.

Theorem 6.1. *Let f be an arbitrary two-party functionality. Assume the existence of:*

- *A two-round, 2-client, m-server pairwise verifiable MPC protocol $\Phi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ for computing f against t server corruptions (see Definition 5.2).*
- *A two-round semi-honest protocol $\Pi_i = (\Pi_{i,1}, \Pi_{i,2}, \mathsf{out}_{\Pi_i})$ for each $i \in [m]$ (see Definition 4.1) where Π_i computes the function $\mathsf{Eval}(i, \cdot)$.*

Then, there exists a NISC protocol Γ for computing f that makes black-box use of $\{\Pi_i\}_{i\in[n]}$ and is secure against static, malicious adversaries in the random oracle model. The communication and computation costs of the protocol are $\mathsf{poly}(\lambda, |f|)$, where $|f|$ denotes the size of the circuit computing f.

Instantiating the pairwise verifiable MPC protocol from Theorem 5.4, we get the following corollary.

Corollary 6.2. *Let f be an arbitrary two-party functionality. There exists a two-round protocol Γ for computing f that makes black-box use of $\{\Pi_i\}_{i\in[n]}$ and is secure against static, malicious* *adversaries in the random oracle model. The communication and computation costs of the protocol are* $\mathsf{poly}(\lambda, |f|)$*, where* $|f|$ *denotes the size of the circuit computing* f*.*

In Section 6.1.1, we describe the construction of the above malicious-secure protocol and in section 6.1.2, we give the proof of security.

6.1.1 Construction

We start with the description of the building blocks used in the construction.

Building Blocks. The construction makes use of the following building blocks.

1. A protocol $\Phi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ that is a two-round, 2-client, m-server pairwise verifiable MPC protocol w.r.t. predicate P for computing the function f against t server corruptions (see Definition 5.2). We set $t = 4\lambda$ and $m = 6t + 1$.

2. An two-round semi-honest inner protocol $\Pi_i = (\Pi_{i,1}, \Pi_{i,2}, \mathsf{out}_{\Pi_i})$ for each $i \in [m]$ (see Definition 4.1) where Π_i computes the function $\mathsf{Eval}(i, \cdot)$ (i.e., the function computed by the i-th server).

3. A non-interactive, straight-line extractable commitment $(\mathsf{Com}, \mathsf{Open})$. Such a commitment scheme can be constructed unconditionally in the random oracle model (see Section 4.3).

4. Two hash functions $H_1 : \{0,1\}^* \to \{0,1\}^\lambda$ and $H_2 : \{0,1\}^* \to \mathcal{S}_{m,\lambda}$ that are modelled as random oracles where $\mathcal{S}_{m,\lambda}$ is the set of all subsets of $[m]$ of size λ.

Description of the Protocol. Let P_0 be the receiver that has private input x_0 and P_1 be the sender that has private input x_1. The common input to both parties is a description of a two-party function f. We give the formal description of a two-round, malicious-secure protocol for computing f in Figure 1.

6.1.2 Proof of Security

Let $\mathcal{A}$ be the malicious adversary that is corrupting either P_0 or P_1. We start with the description of the simulator Sim. Let P_i be the honest client.

Description of Sim**.**

1. **Interaction with the Environment.** For every input value corresponding to the corrupted P_{1-i} that Sim receives from the environment, it writes these values to the input tape of the adversary $\mathcal{A}$. Similarly, the contents of the output tape of $\mathcal{A}$ is written to Sim's output tape.

2. Sim chooses uniform subset K_i of size λ and programs the random oracle H_2 to output this set when queried on the message generated by P_i.

3. Sim starts interacting with the simulator Sim_Φ for the outer protocol by corrupting the client P_{1-i} and the set of servers indexed by K_i. It obtains the first round messages $\{x^i_j\}_{j \in K_i}$ sent by the honest client P_i to the corrupted servers.

- **Round 1:** The receiver P_0 does the following:
  1. It computes $(x_1^0, \ldots, x_m^0, vk_0) \leftarrow \mathsf{Share}(1^\lambda, 0, x_0)$.
  2. For each $j \in [m]$,
     (a) It computes $r_j^0 := H_1(0, j, x_j^0, s_j^0)$ for uniformly chosen $s_j^0 \leftarrow \{0,1\}^\lambda$.
     (b) It computes $\mathsf{com}_j^0 \leftarrow \mathsf{Com}((x_j^0, s_j^0))$.
     (c) It computes $(\pi_{j,1}, sk_j) \leftarrow \Pi_{j,1}(1^\lambda, x_j^0; r_j^0)$.
  3. It computes $K_0 = H_2(0, \{\mathsf{com}_j^0, \pi_{j,1}\}_{j\in[m]}, \mathsf{tag}_0)$ where $\mathsf{tag}_0 \leftarrow \{0,1\}^\lambda$.
  4. It sends $\{\mathsf{com}_j^0, \pi_{j,1}\}_{j\in[m]}$, tag_0, and $\{(x_j^0, s_j^0), \mathsf{Open}(\mathsf{com}_j^0)\}_{j\in K_0}$.
- **Round-2:** The sender does the following:
  1. It runs $\mathsf{chkConsistency}(0, \mathbb{T})$ where $\mathsf{chkConsistency}$ is described in Figure 2 and $\mathbb{T}$ is the transcript in the first round. If $\mathsf{chkConsistency}$ outputs 0, then it aborts.
  2. Else, it computes $(x_1^1, \ldots, x_m^1, vk_1) \leftarrow \mathsf{Share}(1^\lambda, 1, x_1)$.
  1. For each $j \in [m]$,
     (a) It computes $r_j^1 := H_1(1, j, x_j^1, s_j^1)$ for uniformly chosen $s_j^1 \leftarrow \{0,1\}^\lambda$.
     (b) It computes $\mathsf{com}_j^1 \leftarrow \mathsf{Com}((x_j^1, s_j^1))$.
     (c) It computes $\pi_{j,2} \leftarrow \Pi_{j,2}(1^\lambda, x_j^1, \pi_{j,1}; r_j^1)$.
  2. It computes $K_1 = H_2(1, \{\mathsf{com}_j^1, \pi_{j,2}\}_{j\in[m]}, \mathsf{tag}_1)$ where $\mathsf{tag}_1 \leftarrow \{0,1\}^\lambda$.
  3. It sends $\{\mathsf{com}_j^1, \pi_{j,2}\}_{j\in[m]}$, tag_1, and $\{(x_j^1, s_j^1), \mathsf{Open}(\mathsf{com}_j^1)\}_{j\in K_1}$.
- **Output:** To compute the output, the receiver does the following:
  1. It runs $\mathsf{chkConsistency}(1, \mathbb{T})$ where $\mathbb{T}$ is the transcript in the first two rounds. If $\mathsf{chkConsistency}$ outputs 0, then it aborts and outputs $\perp$.
  2. For each $j \in [m]$,
     (a) It runs $\mathsf{out}_{\Pi_j}(\pi_{j,2}, sk_j)$ to obtain ϕ_j.
  3. It runs $\mathsf{Dec}(0, vk_0, \phi_1, \ldots, \phi_m)$ and outputs whatever Dec outputs.

Figure 1: Description of Two-round Malicious 2PC

4. For each $j \in K_i$, it uses the the input x_j^i and uniformly chosen s_j^i to generate the messages in the protocol Π_j as described in Figure 1. For each $j \notin K_i$, it runs the simulator for the inner protocol Π_j to generate the messages on behalf of P_i. To generate the commitments, for each $j \in K_i$, it uses (x_j^i, s_j^i) to compute com_j^i. However, for each $j \notin K_i$, it commits to some dummy values.

5. For each of the unique random oracle queries made by $\mathcal{A}$, Sim samples a uniform element in the range of the oracle and outputs it as the response. Each time Sim generates query to the random oracle on behalf of honest P_i, Sim checks if adversary has already made that query. If that is the case, then it aborts the execution and outputs a special symbol ABORT.

6. On obtaining the protocol message from $\mathcal{A}$, Sim uses the straight-line extractor for the extractable commitment Com and obtains $(x_1^{1-i}, s_1^{1-i}), \ldots, (x_m^{1-i}, s_m^{1-i})$ from $\mathsf{com}_1^{1-i}, \ldots, \mathsf{com}_m^{1-i}$ respectively.

7. It initializes two empty sets I_1 and I_2.

Input: A party index $i \in \{0, 1\}$ and the transcript $\mathbb{T}$.

1. Compute K_i from the transcript $\mathbb{T}$ and the hash function H_2.
2. For each $j \in K_i$,
   (a) It obtains $\{(x^i_j, s^i_j), \mathsf{Open}(\mathsf{com}^i_j)\}$ from $\mathbb{T}$.
   (b) It checks if $\mathsf{Open}(\mathsf{com}^i_j)$ is valid.
   (c) It then checks if $(x^i_j, H_1(i, j, x^i_j, s^i_j))$ is a valid (input,randomness) pair for the protocol Π_j consistent with the transcript $\mathbb{T}$.
   (d) For each $j' \in K_i$, it checks if $P(i, j, j', x^i_j, x^i_{j'}) = 1$.
3. If any of the checks fail, it outputs 0. Else, if all the checks pass, it outputs 1.

Figure 2: Description of $\mathsf{chkConsistency}$

8. For each $j \in [m]$, if $(x^{1-i}_j, H_1(1-i, j, x^{1-i}_j, s^{1-i}_j))$ is not a valid (input,randomness) pair for the protocol Π_j w.r.t. the messages sent by $\mathcal{A}$, then it adds j to the set I_1. It adaptively corrupts the server j in the outer protocol and obtains x^i_j. It uses this as the input to compute the second round message of the protocol Π_j when $i = 1$.

9. It constructs an inconsistency graph G where the vertices correspond to $[m]$ and it adds an edge between j and k if $P(1-i, j, k, x^{1-i}_j, x^{1-i}_k) = 0$. It then computes a 2-approximation for the minimum vertex cover in this graph and calls this vertex cover as I_2. For each $j \in I_2$, it adaptively corrupts the server j in the outer protocol and obtains x^i_j. It uses this as the input to generate the second round message of the protocol Π_j when $i = 1$.

10. If $|I_1| \geq \lambda$ or if $|I_2| \geq \lambda$, then it sends $\perp$ to its ideal functionality.

11. It completes the interaction with $\mathcal{A}$ and if at any point of time, $\mathcal{A}$'s messages do not pass $\mathsf{chkConsistency}$ then Sim sends $\perp$ to the trusted functionality.

12. It provides $\{x^{1-i}_j\}_{j \notin I_1 \cup I_2 \cup K_i}$ to Sim_Φ as the messages sent by the adversary to the honest servers. Sim_Φ queries the ideal functionality on an input x_{1-i} and Sim forwards this to its trusted functionality.

13. If $i = 0$, then if Sim_Φ instructs the ideal functionality to deliver the output to honest P_0, then Sim forwards this message. Otherwise, if Sim_Φ instructs the ideal functionality to deliver $\perp$, Sim sends $\perp$ to the ideal functionality.

14. If $i = 1$, then Sim obtains $z = f(x_0, x_1)$ from the ideal functionality and forwards this to Sim_Φ. Sim_Φ sends the second round protocol messages $\{\phi_j\}_{j \notin I_1 \cup I_2 \cup K_1}$ from the honest servers. For each $j \notin I_1 \cup I_2 \cup K_1$, Sim uses ϕ_j as the output of Π_j and gives this as input to the simulator for Π_j along with $(x^0_j, H_1(0, j, x^0_j, s^0_j))$ as the (input, randomness) pair. We get the final round message for Π_j for each $j \notin I_1 \cup I_2 \cup K_1$ from the inner protocol simulators and we use this to generate the final round message in the protocol.

Proof of Indistinguishability. We now argue that the real execution and the ideal execution are computationally indistinguishable via a hybrid argument.

- Real : This corresponds to the output of the real execution of the protocol.
- Hyb_0 : This hybrid corresponds to the distribution where the random oracle queries of the adversary are answered with a uniformly chosen random element from the image of the oracle. Further, if the adversary makes any queries to the hash functions H_1, H_2 before the exact same query was made by the honest party, we abort. We note that since each query made to the hash functions H_1, H_2 has a component which is a uniformly chosen random string of length λ, the probability that an adversary is able to make a query that exactly matches this string queried by an honest party is $q \cdot 2^{-\lambda}$ (where q is the total number of queries made by the adversary to the random oracles). Hence, this hybrid is statistically close to the previous one.
- Hyb_1 : In this hybrid, we make the following changes:
  1. We use the extractor for the extractable commitment Com to obtain $(x_1^{1-i}, s_1^{1-i}), \ldots,$ (x_m^{1-i}, s_m^{1-i}) from $\mathsf{com}_1^{1-i}, \ldots, \mathsf{com}_m^{1-i}$ respectively.
  2. We construct the sets I_1 and I_2 as described in the simulation.
  3. If $|I_1| \geq \lambda$ or $|I_2| \geq \lambda$, we abort the execution and instruct the honest party to output $\perp$.
  4. If $i = 0$ and if $|I_1| < \lambda$ and $|I_2| < \lambda$, then for each $j \in I_1 \cup I_2 \cup K_i$, we set ϕ_j to be some default value and compute the output of honest P_0.

  In Lemma 6.3, we show that Hyb_0 and Hyb_1 are statistically indistinguishable from the error correction properties of Φ (see Definition 5.1).
- Hyb_2 : In this hybrid, we make the following changes:
  1. We sample a uniform subset K_i (of size λ) and program the random oracle H_2 to output this set when queried on the messages generated by P_i.
  2. For each $j \notin K_i$, we change the commitments com_j^i to be commitments to some dummy values instead of (x_j^i, s_j^i).

  This hybrid is computationally indistinguishable to the previous hybrid from the hiding property of the non-interactive commitment scheme.
- Hyb_3 : In this hybrid, we do the following:
  1. We choose uniform subset K_i of $[m]$ of size λ and program the random oracle H_2 to output this set when queried on the messages generated by P_i.
  2. For each $j \notin K_i$, we run the simulator for the inner protocol and generate the messages from P_i for the protocol Π_j using this simulator.
  3. We compute the sets I_1 and I_2 as before.
  4. If some $j \notin K_i$ is added to I_1 or I_2 and if $i = 1$, we use x_j^i to compute the second round sender message.
  5. If $|I_1| \geq \lambda$ or if $|I_2| \geq \lambda$, we abort as in the previous hybrid.
  6. For $j \notin K_i \cup I_1 \cup I_2$, we use the input x_j^{1-i} extracted from the extractable commitment to compute $\phi_j = \mathsf{Eval}(1^\lambda, j, x_j^0, x_j^1)$.
  7. If $i = 0$, for each $j \in K_i \cup I_1 \cup I_2$, we set ϕ_j to be a default value and use these values instead to compute the output of the receiver P_0.

8. If $i = 1$, then for each $j \notin K_1 \cup I_1 \cup I_2$, we send the input x_j^0, randomness $H_1(0, j, x_j^0, s_j^0)$ and the output ϕ_j to the simulator for Π_j and obtain the final round message in Π_j. We use this to generate the final round message in the overall protocol.

In Lemma 6.4, we show that $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$ from the semi-honest sender security of the inner protocol.

- $\underline{\mathsf{Hyb}_4}$: In this hybrid, we make the following changes:
  1. We (adaptively) corrupt the set of servers corresponding to the indices $K_i \cup I_1 \cup I_2$ and the client P_{1-i}. We run the simulator Sim_Φ for the outer protocol and obtain the first round messages sent by the honest client to these corrupted servers. We use this to complete the execution with $\mathcal{A}$.
  2. We provide $\{x_j^{1-i}\}_{j \notin K_i \cup I_1 \cup I_2}$ (extracted from the extractable commitment) to Sim_Φ as the messages sent by the adversary to the honest servers. Sim_Φ queries the ideal functionality on an input x_{1-i}.
  3. If $i = 0$ then if Sim_Φ instructs the ideal functionality to deliver the output to honest P_0, then we instruct P_0 to output $f(x_0, x_1)$. Otherwise, if Sim_Φ instructs the ideal functionality to deliver $\perp$, we instruct P_0 to output $\perp$.
  4. If $i = 1$, we compute $z = f(x_0, x_1)$ and send this to Sim_Φ as the output from the ideal functionality. Sim_Φ sends the second round protocol messages $\{\phi_j\}_{j \notin K_i \cup I_1 \cup I_2}$ from the honest servers. We use this to generate the final round message of the protocol as in the previous hybrid.

  In Lemma 6.5, we show that $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$ from the security of the outer protocol. We note that output of Hyb_4 is identically distributed to the output of the ideal execution with Sim.

Lemma 6.3. *Assuming the error correction properties of Φ, we have $\mathsf{Hyb}_0 \approx_s \mathsf{Hyb}_1$.*

Proof. We show that if $|I_1| \geq \lambda$ or if $|I_2| \geq \lambda$ then the honest client in Hyb_0 also aborts with overwhelming probability.

- **Case-1: $|I_1| \geq \lambda$:** Note that K_{1-i} is chosen by the random oracle after the adversary generates the message on behalf of the corrupted party in the protocol. We show that since K_{1-i} is uniformly chosen random subset of $[m]$ of size λ, the probability that $|I_1 \cap K_{1-i}| = 0$ is $2^{-O(\lambda)}$. Note that if this event doesn't happen, then the honest client P_i aborts in Hyb_0.

$$\begin{aligned}
\Pr[|K_{1-i} \cap I_1| = 0] &\leq \frac{\binom{m-\lambda}{\lambda}}{\binom{m}{\lambda}} \\
&= \left(1 - \frac{\lambda}{m}\right)\left(1 - \frac{\lambda}{(m-1)}\right) \cdots \left(1 - \frac{\lambda}{(m-(\lambda-1))}\right) \\
&< \left(1 - \frac{\lambda}{m}\right)^{\lambda} < e^{-O(\lambda)}.
\end{aligned}$$

  where the last inequality follows since $m = O(\lambda)$. By an union bound over the set of all the q queries that adversary makes to the random oracle H_2, the probability that there exists some K_{1-i} which is the response of the RO such that $|K_{1-i} \cap I_1| = 0$ is upper bounded by $q \cdot e^{-O(\lambda)}$.
- **Case-2: $|I_2| \geq \lambda$:** Since $|I_2| \geq \lambda$, the size of the minimum vertex cover is at least $\lambda/2$. This means that in the inconsistency graph, there exists a maximum matching of size at least $\lambda/4$.

Let M be the set of vertices for this matching. Note that K_{1-i} is uniformly chosen random subset of $[m]$ of size λ. If any edge of this matching is present in K_{1-i}, then the honest client P_i aborts in Hyb_0. [IKOS07, Theorem 4.1] shows that probability that no edge of this matching is present in K_{1-i} is $2^{-O(\lambda)}$. Again, by an union bound over the set of all the q queries that adversary makes to the random oracle H_2, the probability that there exists some K_{1-i} which is the response of the RO such that no edge in M is in K_{1-i} is upper bounded by $q \cdot 2^{-O(\lambda)}$.

In the case, where $|I_1| \leq \lambda$ and $|I_2| \leq \lambda$, consider an admissible adversary $\mathcal{A}'$ against the protocol Φ that corrupts the set of servers indexed by $I_1 \cup I_2 \cup K_i$. By definition for every server $j, k \notin I_1 \cup I_2 \cup K_i$, it follows that $P(1-i, j, k, x_j^{1-i}, x_k^{1-i}) = 1$. Thus, it follows from the error correction property of Φ that $\mathsf{Hyb}_2 \approx_s \mathsf{Hyb}_3$. $\square$

Lemma 6.4. *Assuming the semi-honest security of the inner protocol, we have that* $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$.

Proof. We sample a uniform subset K_i of $[m]$ of size λ and program the random oracle H_2 to output this set when queried on the messages generated by P_i.

Let $I = [m] \setminus K_i$. We consider a sequence of $|I|$ hybrids between Hyb_2 and Hyb_3 where we change from real to simulated executions of the inner protocol for each $j \in I$ one by one. If Hyb_2 and Hyb_3 are computationally distinguishable, then by a standard hybrid argument, there exists two sub-hybrids $\mathsf{Hyb}_{2,j-1}$ and $\mathsf{Hyb}_{2,j}$ which differ only in the j-th execution and are computationally distinguishable. Specifically, in $\mathsf{Hyb}_{2,j}$, the messages in the protocol Π_j is generated as in the ideal execution and in the $\mathsf{Hyb}_{2,j-1}$ it is generated as in the real execution. We now show that this contradicts the semi-honest security of the inner protocol.

We begin interacting with external challenger and provide x_j^i as the input used by P_i in Π_j. Amongst all the queries made by $\mathcal{A}$ to the random oracle H_1 where the first two inputs are $(1-i, j)$, we choose one of these queries $(1-i, j, x_j^{1-i}, s_j^{1-i})$ at random and give x_j^{1-i} as the input of the corrupted party. The challenger provides with a random tape r_j^{1-i} to be used by P_{1-i}. We provide r_j^{1-i} as the response from the random oracle. On receiving the protocol message from $\mathcal{A}$, we run the extractor for the extractable commitment Com on com_j^{1-i} and obtain $(\overline{x}_j^{1-i}, \overline{s}_j^{1-i})$. We consider the following cases.

1. If j is added to I_1 or I_2 then:
   - If $i = 1$, we use x_j^i to generate the second round sender message. We generate the view of the adversary and run the distinguisher between $\mathsf{Hyb}_{2,j}$ and $\mathsf{Hyb}_{2,j-1}$ on this view and output whatever it outputs.
   - If $i = 0$, we set ϕ_j to be an arbitrary value and generate the view of the adversary and the output of the honest party as before. We run the distinguisher between $\mathsf{Hyb}_{2,j}$ and $\mathsf{Hyb}_{2,j-1}$ on these values and output whatever it outputs.

2. If j is not added to I_1 or I_2 but $(\overline{x}_j^{1-i}, \overline{s}_j^{1-i}) \neq (x_j^{1-i}, s_j^{1-i})$, then we output a random bit to the external challenger.

3. If j is not added to I_1 or I_2 and $(\overline{x}_j^{1-i}, \overline{s}_j^{1-i}) = (x_j^{1-i}, s_j^{1-i})$, then we continue with the rest of the execution using the messages from the challenger ($i = 1$) or the output from the challenger ($i = 0$) to compute the view of the adversary and output of the honest party. We run the distinguisher between $\mathsf{Hyb}_{2,j}$ and $\mathsf{Hyb}_{2,j-1}$ and output whatever it outputs.

We note that if j is not added to I_1 or I_2 and $(\overline{x}_j^{1-i}, \overline{s}_j^{1-i}) = (x_j^{1-i}, s_j^{1-i})$, then the input to the distinguisher is identical to $\mathsf{Hyb}_{2,j-1}$ if the challenger generated the messages of Π_j as in the real execution and otherwise, it is identical to $\mathsf{Hyb}_{2,j}$. Similarly, if j is added to I_1 or I_2, then the input to the distinguisher is identical to $\mathsf{Hyb}_{2,j-1}$ if the challenger generated the messages of Π_j as in the real execution and otherwise, it is identical to $\mathsf{Hyb}_{2,j}$.

Finally, conditioning on j not added to I_1 or I_2, the probability that $(\overline{x}_j^{1-i}, \overline{s}_j^{1-i}) \neq (x_j^{1-i}, s_j^{1-i})$ is at least $1-1/q-\mathsf{negl}(\lambda)$ (and at most $1-1/q+\mathsf{negl}(\lambda)$) where q is the total number of queries made by the adversary to the random oracle H_1. Let us assume that the probability that the distinguisher correctly predicts whether it is given a sample from $\mathsf{Hyb}_{2,j}$ and $\mathsf{Hyb}_{2,j-1}$ to be $1/2+\mu(\lambda)$ (for some non-negligible $\mu(\lambda)$). Let ϵ be the probability that j is added to I_1 or I_2. Let p be the probability that the above reduction correctly predicts whether it is interacting with the real execution or the ideal execution. Then,

$$\begin{aligned} p &\geq (1/2+\mu(\lambda))\epsilon + (1-\epsilon)((1-1/q-\mathsf{negl}(\lambda))(1/2) + (1/q-\mathsf{negl}(\lambda))(1/2+\mu(\lambda))) \\ &\geq (1/2+\mu(\lambda))\epsilon + (1-\epsilon)(1/2+\mu(\lambda)/q) - \mathsf{negl}(\lambda) \\ &\geq 1/2+\mu(\lambda)/q + \epsilon(\mu(\lambda)-\mu(\lambda)/q) - \mathsf{negl}(\lambda) \\ &\geq 1/2+\mu(\lambda)/q - \mathsf{negl}(\lambda) \end{aligned}$$

and this contradicts the semi-honest security of the inner protocol. □

Lemma 6.5. *Assuming the security of the outer protocol Φ, we have $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$.*

Proof. Assume for the sake of contradiction that Hyb_3 and Hyb_4 are computationally distinguishable. We give a reduction to breaking the security of the outer protocol.

We begin interacting with the external challenger by providing the input x_i of the honest client P_i. We then corrupt the other client P_{1-i} and the set of servers indexed by K_i. We obtain the first round messages sent from the honest client P_i to the corrupted servers and we begin interacting with $\mathcal{A}$ using these messages. For each server that is added to I_1 or I_2, we adaptively corrupt that server and obtain the first round message sent from the honest client to this server. We use this message to continue with the rest of the execution as in Hyb_3. At the end of the protocol execution, we send $\{x_j^{1-i}\}_{j \notin K_i \cup I_1 \cup I_2}$ as the first round messages sent by the corrupted client P_{1-i} to the honest servers. If P_0 is uncorrupted, we send $\{\phi_j\}_{j \in K_i \cup I_1 \cup I_2}$ (set to be arbitrary values as in Hyb_3) to the challenger and it provides the output of P_0 and we instruct P_0 to output the same. If P_0 is corrupted, we obtain $\{\phi_j\}_{j \notin K_i \cup I_1 \cup I_2}$ from the external challenger and we use this to generate the final round message in the protocol. We finally run the distinguisher between Hyb_3 and Hyb_4 on the view of $\mathcal{A}$ and the output of P_0 (if it is uncorrupted) and output whatever the distinguisher outputs.

The above reduction emulates an admissible adversary as by definition the first round message sent to the honest servers pass the pairwise verification w.r.t. predicate P. Since $|K_i \cup I_1 \cup I_2| \leq |K_i| + |I_1| + |I_2| = 3\lambda = t$, the reduction emulates an admissible adversary that corrupts at most t servers. Thus, if the messages generated by the external challenger are done as in the real execution then input to the distinguisher is identical to Hyb_3. Else, it is identically distributed to Hyb_4. This implies that the reduction breaks the security of the protocol Φ and this is a contradiction. □

Protocols for (Batch) OT and OLE. To construct OT or OLE protocols with malicious security, we instantiate the outer protocol with a simplified variant of the pairwise verifiable protocol from Section 5.2.1. We instantiate the inner protocol with a 2-round semi-honest protocol emulating the server computations in the outer protocol by making parallel calls to OT/OLE. The latter can be obtained using information-theoretic Boolean/arithmetic variants of Yao's protocol that efficiently apply to "simple" (e.g., log-depth) functionalities. This gives us the black-box feasibility result of constructing 2-round malicious OT/OLE from 2-round semi-honest OT/OLE.

We now sketch the details of a "constant-rate" variant of the above blueprint. We follow the high level approach of the constant-rate (multi-round) protocols from [IPS08, IPS09], except for using a 2-round outer protocol based on bivariate polynomials (or tensored AG codes) instead of an outer protocol based on univariate polynomials (or AG codes).

In more detail, we instantiate the above compiler with the following building blocks. Consider first the case of batch-OLE over a big field $\mathbb{F}$ ($|\mathbb{F}| \geq 2^\lambda$). Here we use a pairwise verifiable outer protocol computing $\Omega(n^2)$ instances of OLE using packed bivariate Shamir secret sharing over $\mathbb{F}$, namely where a single bivariate polynomial encodes an $\ell \times \ell$ matrix of secrets for $\ell = \Omega(n)$. The local computations done by the n servers in the outer protocol consist of just $O(n^2)$ parallel computations of the form $y_i = a_i x_i + b_i$, or a constant number of such computations per OLE instance. These computations require the inner protocol to make $O(1)$ semi-honest OLE calls per malicious OLE instance. The authentication of the outputs of the outer protocol can be achieved with a constant overhead by either appending a simple arithmetic MAC to the y_i, as in the protocol from Section 5.2.1, or by using the error-correcting property of the outer protocol (which enables detection of tampering with the output shares of corrupted servers).

For the case of OT, one can use an analogous protocol based on tensored algebraic geometric codes, in which the field size does not grow with the number of servers n. Here the computation of $y_i = a_i x_i + b_i$ has a constant-size circuit, and hence requires the inner protocol to use a constant number of OTs. The output authentication can be achieved with a constant overhead via the error-correction of the outer protocol.

6.2 Protocol Compiler in the OT Correlations Model

In this section, we describe a protocol compiler that transforms two-round semi-malicious two-party protocol to a two-round malicious-secure protocol. This transformation is in the standard 1-out-of-2 OT correlations model. We state the formal theorem below.

Theorem 6.6. *Let f be an arbitrary two-party functionality. Assume the existence of:*

- *A two-round, 2-client, m-server pairwise verifiable MPC protocol $\Phi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ for computing f against t server corruptions (see Definition 5.2).*
- *A two-round semi-malicious protocol $\Pi_i = (\Pi_{i,1}, \Pi_{i,2}, \mathsf{out}_{\Pi_i})$ for each $i \in [m]$ (see Definition 4.3) where Π_i computes the function $\mathsf{Eval}(i, \cdot)$.*

Then, there exists a NISC protocol Γ for computing f that makes black-box use of $\{\Pi_i\}_{i\in[n]}$ and is secure against static, malicious adversaries in the 1-out-of-2 OT correlations model. The communication and computation costs of the protocol are $\mathsf{poly}(\lambda, |f|)$, where $|f|$ denotes the size of the circuit computing f and the size of the OT correlations shared between the parties is a fixed polynomial in the security parameter and is independent of the size of the function f.

Instantiating the pairwise verifiable MPC protocol from Theorem 5.4, we get the following corollary.

Corollary 6.7. *Let f be an arbitrary two-party functionality. There exists a two-round protocol Γ for computing f that makes black-box use of $\{\Pi_i\}_{i\in[n]}$ and is secure against static, malicious* *adversaries in the 1-out-of-2 OT correlations model. The communication and computation costs of the protocol are* $\mathsf{poly}(\lambda, |f|)$*, where* $|f|$ *denotes the size of the circuit computing* f *and the size of the OT correlations shared between the parties is a fixed polynomial in the security parameter and is independent of the size of the function* f*.*

6.2.1 Construction

Our construction makes use of a single round Rabin OT protocol which we describe how to construct in the presence of 1-out-of-2 OT correlations. Here we consider the case of standard (1-sided) NISC for simplicity. An extension to the two-sided case will be discussed in Section 6.3.

Constructing Single Round Rabin OT protocol in OT Correlations Model. To construct a Rabin OT protocol with erasure probability $1 - 1/p$ for some integer p, we do as follows:

1. We first transform the 1-out-2 OT correlations non-interactively to 1-out-of-p correlations. Such a transformation was described in [NP99].

2. We then use the transformation described in [IPS08, Section 2] to convert 1-out-of-p random OT correlations into a single round Rabin OT protocol with erasure probability $1 - 1/p$.

Building Blocks. We start with the description of the building blocks:

1. A protocol $\Phi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ which is a two-round, 2-client, m-server pairwise verifiable protocol w.r.t. predicate P for computing the function f against t server corruptions (see Definition 5.2). We set $t = 4\lambda$ and $m = 7t$.

2. An two-round semi-malicious inner protocol (see Definition 4.3) $\Pi_i = (\Pi_{i,1}, \Pi_{i,2}, \mathsf{out}_\Pi)$ for each $i \in [m]$ where Π_i computes the function $\mathsf{Eval}(1^\lambda, i, \cdot)$ (i.e., the function computed by the i-th server).

3. A single round Rabin OT protocol $\mathsf{RabinOT}$ with erasure probability $1 - \lambda/m$. We extend the syntax of the Rabin OT protocol to take in m strings and each of these strings are independently erased with probability $1 - \lambda/m$.

Description of the Protocol. Let P_0 be the receiver with private input x_0 and P_1 be the sender with private input x_1. The common input to both parties is a description of a function f. The formal description of the protocol appears in Figure 3.

6.2.2 Proof of Security

We start with the description of the simulator.

Description of Sim. Let $\mathcal{A}$ be an adversary that corrupts either P_0 or P_1. We assume that P_i is the honest client.

1. **Interaction with the Environment.** For every input value corresponding to the corrupted P_{1-i} that Sim receives from the environment, it writes these values to the input tape of the adversary $\mathcal{A}$. Similarly, the contents of the output tape of $\mathcal{A}$ is written to Sim's output tape.

- **Round-1:** In the first round, the party P_0 with input x_0 does the following:
  1. It computes $(x_1^0, \ldots, x_m^0, vk_0) \leftarrow \mathsf{Share}(1^\lambda, 0, x_0)$.
  2. It chooses a random string $r_h^0 \leftarrow \{0,1\}^\lambda$ for every $h \in [m]$ and sets $y_h^0 = (r_h^0, x_h^0)$.
  3. It computes $\mathsf{msg}^0 \leftarrow \mathsf{RabinOT}(y_1^0, \ldots, y_m^0)$.
  4. For each $h \in [m]$, it computes $(\pi_{h,1}, sk_h) := \Pi_{h,1}(1^\lambda, x_h^0; r_h^0)$.
  5. It sends $(\{\pi_{h,1}\}_{h\in[m]}, \mathsf{msg}^0)$.
- **Round-2:** In the second round, P_1, with input x_1 does the following:
  1. It computes $(x_1^1, \ldots, x_m^1, vk_1) \leftarrow \mathsf{Share}(1^\lambda, 1, x_1)$.
  2. It chooses a random string $r_h^1 \leftarrow \{0,1\}^\lambda$ for every $h \in [m]$ and sets $y_h^1 = (r_h^1, x_h^1)$.
  3. It computes $\mathsf{msg}^1 \leftarrow \mathsf{RabinOT}(y_1^1, \ldots, y_m^1)$.
  4. It decrypts msg^0 to obtain $\{r_h^0, x_h^0\}_{h\in K_0}$ for some subset K_0 (the rest of the positions are erased).
  5. For each $h \in K_0$, it checks:
     (a) If $\pi_{h,1} := \Pi_{h,1}(1^\lambda, x_h^0; r_h^0)$.
     (b) For each $h' \in K_0$, if $P(0, h, h', x_h^0, x_{h'}^0) = 1$.
  6. If any of the above checks fail, it aborts.
  7. Else, for each $h \in [m]$, it computes $\pi_{h,2} := \Pi_{h,2}(1^\lambda, x_h^1, \pi_{h,1}; r_h^1)$.
  8. It sends $\{\pi_{h,2}\}_{h\in[m]}$ and msg^1 to P_0.
- **Output Computation.** To compute the output, P_i does the following:
  1. It decrypts msg^1 to obtain $\{r_h^1, x_h^1\}_{h\in K_1}$ for some subset K_1 (the rest of the positions are erased).
  2. For each $h \in K_1$, it checks:
     (a) If $\pi_{h,2} := \Pi_{h,2}(1^\lambda, x_h^1, \pi_{h,1}; r_h^1)$.
     (b) For each $h' \in K_1$, if $P(1, h, h', x_h^1, x_{h'}^1) = 1$.
  3. If any of the above checks fail, it aborts.
  4. Else, for every $h \in [m]$, it computes $\phi_h := \mathsf{out}_{\Pi_h}(\pi_{h,2}, sk_h)$.
  5. It computes $z \leftarrow \mathsf{Dec}(0, vk_0, \phi_1, \ldots, \phi_m)$ and outputs z.

Figure 3: Description of the Two-Round 2PC Protocol in the OT Correlations Model

2. Sim samples a subset K_i where each element of $[m]$ is added independently to K_i with probability λ/m. If $|K_i| \geq 2\lambda$, then Sim aborts. Looking ahead, K_i chosen above will be the set of strings that are not erased in msg^i. It sets $\{y_j^i\}_{j\notin K_i}$ to be dummy values.

3. Sim starts interacting with the simulator Sim_Φ for the outer protocol by corrupting the client P_{1-i} and the set of servers indexed by K_i. It obtains the first round messages $\{x_j^i\}_{j\in K_i}$ sent by the honest client to the corrupted servers.

4. For each $j \in K_i$, it uses the the input x_j^i and uniformly chosen r_j^i to generate the messages in the protocol Π_j as described in Figure 3. It computes $y_j^i = (r_j^i, x_j^i)$ for each $j \in K_i$ and computes msg^i. For each $j \notin K_i$, it runs the simulator for the inner protocol Π_j to generate the messages on behalf of P_i.

5. On obtaining the first round message from $\mathcal{A}$, Sim uses the Rabin OT extractor on msg^{1-i}

and obtains $(r_1^{1-i}, x_1^{1-i}), \ldots, (r_m^{1-i}, x_m^{1-i})$. It also samples a subset K_{1-i} where each element from $[m]$ is independently added to K_{1-i} with probability λ/m. It then uses K_{1-i} to perform the same checks done by honest P_i in the protocol.

6. It initializes two empty sets I_1 and I_2.

7. For each $j \in [m]$, if (x_j^{1-i}, r_j^{1-i}) is not a valid (input,randomness) pair for the protocol Π_j w.r.t. the messages received then it adds j to the set I_1. It adaptively corrupts the server j in the outer protocol and obtains x_j^i. It uses this to send the second round message of the protocol Π_j in the case where $i = 1$.

8. It constructs an inconsistency graph G where the vertices correspond to $[m]$ and it adds an edge between j and k if $P(1-i, j, k, x_j^{1-i}, x_k^{1-i}) = 0$. It then computes a 2-approximation for the minimum vertex cover in this graph and calls this vertex cover as I_2. For each $j \in I_2$, it adaptively corrupts the server j in the outer protocol and obtains x_j^i. It uses this to send the second round message of the protocol Π_j in the case where $i = 1$.

9. If $|I_1| \geq \lambda$ or if $|I_2| \geq \lambda$, then it sends $\perp$ to its ideal functionality.

10. It completes the interaction with $\mathcal{A}$ and if at any point of time, $\mathcal{A}$'s messages does not pass the checks described in the protocol then Sim sends $\perp$ to the trusted functionality.

11. It provides $\{x_j^{1-i}\}_{j \notin I_1 \cup I_2 \cup K_i}$ to Sim_Φ as the messages sent by the adversary to the honest servers. Sim_Φ queries the ideal functionality on an input x_{1-i} and Sim forwards this to its trusted functionality.

12. If $i = 0$, then if Sim_Φ instructs the ideal functionality to deliver the output to honest P_0, then Sim forwards this message. Otherwise, if Sim_Φ instructs the ideal functionality to deliver $\perp$, Sim sends $\perp$ to the ideal functionality.

13. If $i = 1$, then Sim obtains $z = f(x_0, x_1)$ from the ideal functionality and forwards this to Sim_Φ. Sim_Φ sends the second round protocol messages $\{\phi_j\}_{j \notin I_1 \cup I_2 \cup K_1}$ from the honest servers. For each $j \notin I_1 \cup I_2 \cup K_1$, Sim uses ϕ_j as the output of Π_j and gives this as input to the simulator for Π_j along with (x_j^0, r_j^0) as the (input, randomness) pair. We get the final round message for Π_j for each $j \notin I_1 \cup I_2 \cup K_1$ from the inner protocol simulators and we use this to generate the final round message in the protocol.

Proof of Indistinguishability. We now argue that the real execution and the ideal execution are computationally indistinguishable via a hybrid argument.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the output of the real execution of the protocol.
- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we make the following changes:
  1. We sample a subset K_i where each element in $[m]$ is independently added to K_i with probability λ/m. If $|K_i| \geq 2\lambda$, we abort.
  2. We use the Rabin OT extractor on msg^{1-i} to obtain $(r_1^{1-i}, x_1^{1-i}), \ldots, (r_m^{1-i}, x_m^{1-i})$
  3. We construct the sets I_1 and I_2 as described in the simulation.
  4. If $|I_1| \geq \lambda$ or $|I_2| \geq \lambda$, we abort the execution and instruct the honest party to abort the execution.

5. If $i = 0$ and if $|I_1| < \lambda$ and $|I_2| < \lambda$, then for each $j \in I_1 \cup I_2 \cup K_i$, we set ϕ_j to be a default value and compute the output of honest P_0.

We show in Lemma 6.8 that Hyb_0 and Hyb_1 are statistically indistinguishable from the error correction properties of Φ.

- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we make the following changes:
  1. We choose a subset K_i as described in the simulation.
  2. For each $j \notin K_i$, we set y_j^i to be dummy values.

  This hybrid is computationally indistinguishable to the previous hybrid from the security of the Rabin-OT protocol.

- $\underline{\mathsf{Hyb}_3}$: In this hybrid, we do the following:
  1. We choose the set K_i as specified before.
  2. For each $j \notin K_i$, we run the simulator for the inner protocol and generate the messages from P_i for the protocol Π_j using this simulator.
  3. We compute the sets I_1 and I_2 as before.
  4. If some $j \notin K_i$ is added to I_1 or I_2 and if $i = 1$, we use x_j^i to compute the second round sender message.
  5. If $|I_1| \geq \lambda$ or if $|I_2| \geq \lambda$, we abort as in the previous hybrid.
  6. For $j \notin K_i \cup I_1 \cup I_2$, we use the input x_j^{1-i} extracted from the Rabin OT protocol to compute $\phi_j = \mathsf{Eval}(1^\lambda, j, x_j^0, x_j^1)$.
  7. If $i = 0$, for each $j \in K_i \cup I_1 \cup I_2$, we set ϕ_j to be a default value and use these values instead to compute the output of the receiver P_0.
  8. If $i = 1$, then for each $j \notin K_1 \cup I_1 \cup I_2$, we send the input x_j^0, randomness r_j^0 and the output ϕ_j to the simulator for Π_j and obtain the final round message in Π_j. We use this to generate the final round message in the overall protocol.

  In Lemma 6.9, we show that $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$ from the semi-malicious security of the inner protocol.

- $\underline{\mathsf{Hyb}_4}$: In this hybrid, we make the following changes:
  1. We (adaptively) corrupt the set of servers corresponding to the indices $K_i \cup I_1 \cup I_2$ and the client P_{1-i}. We run the simulator Sim_Φ for the outer protocol and obtain the first round messages sent by the honest client to these corrupted servers. We use this to complete the execution with $\mathcal{A}$.
  2. We provide $\{x_j^{1-i}\}_{j \notin K_i \cup I_1 \cup I_2}$ (extracted from the extractable commitment) to Sim_Φ as the messages sent by the adversary to the honest servers. Sim_Φ queries the ideal functionality on an input x_{1-i}.
  3. If $i = 0$ then Sim if Sim_Φ instructs the ideal functionality to deliver the output to honest P_0, then we instruct P_0 to output $f(x_0, x_1)$. Otherwise, if Sim_Φ instructs the ideal functionality to deliver $\perp$, we instruct P_0 to output $\perp$.
  4. If $i = 1$, we compute $z = f(x_0, x_1)$ and send this to Sim_Φ as the output from the ideal functionality. Sim_Φ sends the second round protocol messages $\{\phi_j\}_{j \notin K_i \cup I_1 \cup I_2}$ from the honest servers. We use this to generate the final round message of the protocol as in the previous hybrid.

Via a similar proof in Lemma 6.5, we can show that $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$ from the security of the outer protocol. We note that output of Hyb_4 is identically distributed to the output of the ideal execution with Sim.

Lemma 6.8. *Assuming the error correction property of* Φ*, we have* $\mathsf{Hyb}_0 \approx_s \mathsf{Hyb}_1$.

Proof. It follows from standard Chernoff bounds that only with probability $2^{-O(\lambda)}$ that $|K_i| \geq 2\lambda$. We now argue that if $|I_1| \geq \lambda$ or if $|I_2| \geq \lambda$ then with probability at least $1 - 2^{-O(\lambda)}$, the honest client in Hyb_0 aborts.

- **Case-1:** $|I_1| \geq \lambda$**.** Note that if $|K_{1-i} \cap I_1| \neq 0$ then the honest client P_i aborts. Note that each element in $[m]$ is added to K_{1-i} independently with probability λ/m. Thus, the probability that no element in I_1 is added to K_{1-i} is at most $(1 - \frac{\lambda}{m})^{\lambda} \leq 2^{-O(\lambda)}$ (since $m = O(\lambda)$).
- **Case-2:** $|I_2| \geq \lambda$**:** Since $|I_2| \geq \lambda$, the size of the minimum vertex cover is at least $\lambda/2$. This means that in the inconsistency graph there exists a maximum matching of size at least $\lambda/4$ edges. Let M be the set of vertices for this matching. If there exists at least one edge of this matching in K_{1-i} then the honest client aborts in Hyb_0. The probability that no edge of this matching is in K_{1-i} is $(1 - \frac{\lambda^2}{m^2})^{\lambda/4} \leq 2^{-O(\lambda)}$ (since $m = O(\lambda)$).

In the case, where $|I_1| \leq \lambda$ and $|I_2| \leq \lambda$, consider an admissible adversary $\mathcal{A}'$ against the protocol Φ that corrupts the set of servers indexed by $I_1 \cup I_2 \cup K_i$. By definition for every servers $j, k \notin I_1 \cup I_2 \cup K_i$, it follows that $P(1-i, j, k, x_j^{1-i}, x_k^{1-i}) = 1$. Thus, it follows from the error correction property of Φ that $\mathsf{Hyb}_2 \approx_s \mathsf{Hyb}_3$. □

Lemma 6.9. *Assuming the semi-malicious security of the inner protocol, we have* $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$.

Proof. We sample the subset K_i as before.

Let $I = [m] \backslash K_i$. We consider a sequence of $|I|$ hybrids between Hyb_2 and Hyb_3 where we change from real to simulated executions of the inner protocol for each $j \in I$ one by one. If Hyb_2 and Hyb_3 are computationally distinguishable, then by a standard hybrid argument, there exists two sub-hybrids $\mathsf{Hyb}_{2,j-1}$ and $\mathsf{Hyb}_{2,j}$ which differ only in the j-th execution and are computationally distinguishable. Specifically, in $\mathsf{Hyb}_{2,j}$, the messages in the protocol Π_j is generated as in the ideal execution and in the $\mathsf{Hyb}_{2,j-1}$ it is generated as in the real execution. We now show that this contradicts the weak adaptive semi-honest security of the inner protocol.

We begin interacting with external challenger and provide x_j^i as the input used by P_i in Π_j and use the messages received from the challenger to generate the messages in the protocol. On receiving the first round message from $\mathcal{A}$, we run the extractor for the Rabin OT protocol and obtain (x_j^{1-i}, r_j^{1-i}). We compute the sets I_1 and I_2 as before. If $j \notin I_1 \cup I_2$, we send (x_j^{1-i}, r_j^{1-i}) to the challenger and obtain the second round message in case $i = 1$, and otherwise obtain the output ϕ_j in case $i = 0$. If j is added to $I_1 \cup I_2$, then we use x_j^1 to generate the second round message in case $i = 1$ and set ϕ_j to be an arbitrary value in case $i = 0$. We then proceed to compute the view of $\mathcal{A}$ and the output of P_0 (in case $i = 0$) as in $\mathsf{Hyb}_{2,j-1}$ and run the distinguisher between $\mathsf{Hyb}_{2,j-1}$ and $\mathsf{Hyb}_{2,j}$ on this and output whatever it outputs.

We note that the input to the distinguisher is identical to $\mathsf{Hyb}_{2,j-1}$ if the challenger generated the messages of Π_j as in the real execution and otherwise, it is identical to $\mathsf{Hyb}_{2,j}$. Since the distinguisher is assumed to distinguish between $\mathsf{Hyb}_{2,j-1}$ and $\mathsf{Hyb}_{2,j}$ with non-negligible advantage, it contradicts the semi-malicious security of the inner protocol. □

6.3 Extension to the Two-Sided Setting

In this subsection, we explain how to extend the protocol described in Section 6.1 to the bidirectional communication model. Specifically, we want to construct an two-sided NISC protocol where in each round, both parties can send a message and we require both parties get the output at the end of the second round. The extension for the protocol in the OT correlations model is similar.

Construction. The construction is very similar to the one described in Figure 1 except that we run two instances of the inner protocol for each $j \in [m]$, namely, Π_j^0 and Π_j^1 where the parties use the same input in both the executions (but use independently chosen randomness). Here, Π_j^0 is the protocol that delivers output to P_0 and Π_j^1 is the protocol that delivers output to P_1. Additionally, for each $j \in [m]$, the parties send an extractable commitment to the input and the random strings used in Π_j^0 and Π_j^1 respectively. In each round $u \in [2]$, the parties use the random oracle H_2 to derive a set K_0^u, K_1^u respectively as in the previous protocol description. The party P_i (for each $i \in \{1, 2\}$) then opens the above generated extractable commitment for those executions indexed by K_i^u. The $\mathsf{chkConsistency}$ run by P_i is modified so that it checks if the input, randomness pair is consistent in Π_j^0 and Π_j^1 for each $j \in K_{1-i}^u$. The output computation by both parties is done exactly as described in Figure 1.

Proof of Indistinguishability. Since the protocol is symmetric, we assume without loss of generality that P_0 is the honest client. We now argue that the real execution and the ideal execution are computationally indistinguishable.

- $\underline{\mathsf{Real}}$: This corresponds to the output of the real execution of the protocol.
- $\underline{\mathsf{Hyb}_0}$: This hybrid corresponds to the distribution where the adversaries random oracle queries are answered with uniformly chosen random elements from the image of the oracle. Further, if the adversary makes any queries to the hash functions H_1, H_2 before the exact same query was made by the honest party, we abort. Via an identical argument made in Section 6.1.2, we note that this hybrid is statistically close to the previous hybrid.
- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we make the following changes:
  1. Let K_0 be the union of K_0^1 and K_0^2.
  2. We use the extractor for the extractable commitment Com to obtain $(x_1^1, s_1^{1,0}, s_1^{1,1}), \ldots,$ $(x_m^1, s_m^{1,0}, s_m^{1,1})$ from $\mathsf{com}_1^1, \ldots, \mathsf{com}_m^1$.
  3. We construct the sets I_1 as described in the simulation and I_2 exactly as before.
  4. If $|I_1| \geq \lambda$ or $|I_2| \geq \lambda$, we abort the execution and instruct the honest party to abort the execution.
  5. If $|I_1| < \lambda$ and $|I_2| < \lambda$, then for each $j \in I_1 \cup I_2 \cup K_0$, we set ϕ_j to be a default value and compute the output of honest P_0.

  Again, via an identical argument to Lemma 6.3, we can show that Hyb_0 and Hyb_1 are statistically close from the error correction properties of Φ.
- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we make the following changes:
  1. We choose uniform subsets K_0^1 and K_0^2 (of size λ) and set K_0 to be the union of these sets. We program the random oracle H_2 to output the appropriate set when queried on the messages generated by P_0.

2. For each $j \notin K_0$, we change the commitments com_j^0 to be commitments to some dummy values.

This hybrid is computationally indistinguishable to the previous hybrid from the hiding property of the non-interactive commitment scheme.

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

1. For each round $u \in [2]$, we choose uniform subsets K_0^u (of size λ) and set K_0 to be the union of all these sets. We program the random oracle H_2 to output the appropriate set when queried on the messages generated by P_0.

2. For each $j \notin K_0$, we run the simulator for the inner protocols for both Π_j^0 and Π_j^1 to generate the messages from P_0.

3. If some $j \notin K_0$ is added to I_1 or I_2, then we use x_j^0 as the private input to generate the second round message in Π_j^1.

4. In any round, if $|I_1| \geq \lambda$ or $|I_2| \geq \lambda$, we abort as in the previous hybrid.

5. For $j \notin I_1 \cup I_2 \cup K_0$, we use the input x_j^1 extracted from the extractable commitment to compute $\phi_j = \mathsf{Eval}(1^\lambda, j, x_j^0, x_j^1)$. We send the input x_j^1, randomness $H_1(1, j, x_j^1, s_j^{1,1})$ and the output ϕ_j to the simulator for Π_j^1 and obtain the final round message in Π_j^1. We use this to generate the final round message to be sent by P_0 in the protocol Π_j^1.

6. To compute the output, we do the following. For $j \notin I_1 \cup I_2 \cup K_0$, we use the input x_j^1 extracted from the extractable commitment to compute $\phi_j = \mathsf{Eval}(1^\lambda, j, x_j^0, x_j^1)$. For each $j \in I_1 \cup I_2 \cup K_0$, we set ϕ_j to be a default value. We use this to compute the output as in the previous hybrid.

We can use the argument described in Lemma 6.4 for the case where P_0 is uncorrupted, to replace all $\{\Pi_j^0\}_{j \notin K_0}$ to the simulated distribution. We then use a similar argument described in Lemma 6.4 for the case where P_0 is corrupted to replace all $\{\Pi_j^1\}_{j \notin K_1}$ to the simulated distribution.

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

1. We (adaptively) corrupt the set of servers corresponding to the indices $I_1 \cup I_2 \cup K_0$ and the client P_1. We run the simulator Sim_Φ for the outer protocol and obtain the first round messages sent by the honest client to these corrupted servers. We use this to complete the execution with $\mathcal{A}$.

2. We provide $\{x_j^1\}_{j \notin I_1 \cup I_2 \cup K_0}$ (extracted from the extractable commitment) to Sim_Φ as the messages sent by the adversary to the honest servers. Sim_Φ queries the ideal functionality on an input x_1 and we forward this to the ideal functionality.

3. The ideal functionality outputs $z = f(x_0, x_1)$ and we send this to Sim_Φ. Sim_Φ sends the second round protocol messages $\{\phi_j\}_{j \notin I_1 \cup I_2 \cup K_0}$ from the honest servers. We use this to generate the final round message of the protocol $\{\Pi_j^1\}_{j \notin I_1 \cup I_2 \cup K_0}$ as in the previous hybrid.

4. On obtaining the final round message from the adversary, we update the set I_1 and adaptively corrupt those servers to obtain the first round message from P_0 to those servers. If Sim_Φ instructs the ideal functionality to deliver the output then we instruct P_0 to output $f(x_0, x_1)$. Otherwise, we instruct it to output $\perp$.

In Lemma 6.10, we show that $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$ from the security of the outer protocol. We note that output of Hyb_4 is identically distributed to the output of the ideal execution with the simulator.

Lemma 6.10. *Assuming the security of the outer protocol Φ, we have $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$.*

Proof. Assume for the sake of contradiction that Hyb_3 and Hyb_4 are computationally distinguishable. We give a reduction to breaking the security of the outer protocol.

We begin interacting with the external challenger by providing the input x_0 of the honest client P_0. We then corrupt the other client P_1 and the set of servers indexed by K_0. We obtain the first round messages sent from the honest client P_i to the corrupted servers and we begin interacting with $\mathcal{A}$ using these messages. For each server that is added to I_1 or I_2, we adaptively corrupt that server and obtain the first round message sent from the honest client to this server. We use this message to continue with the rest of the execution as described in Hyb_3. Before sending the final round message on behalf of P_0, we send $\{x_j^1\}_{j \notin I_1 \cup I_2 \cup K_0}$ as the first round messages sent by the corrupted client P_1 to the honest servers.

The challenger replies with $\{\phi_j\}_{j \notin I_1 \cup I_2 \cup K_0}$ as the final round message from the honest servers sent to the corrupt client. We use this as the protocol output to run the inner protocol simulator for $\{\Pi_j^1\}_{j \notin I_1 \cup I_2 \cup K_0}$ (along with the consistent input and randomness used by adversary) to generate the final round messages from P_0. On receiving the final round message from the adversary, we update the set I_1 as described in the simulation and adaptively corrupt the newly added servers to I_1. We obtain the first round message sent by the honest client P_0 to these servers. The challenger provides the output of P_0 and we instruct P_0 to output the same. We finally run the distinguisher between Hyb_3 and Hyb_4 on the view of $\mathcal{A}$ and the output of P_0 and output whatever the distinguisher outputs.

We observe that the reduction emulates an admissible adversary as the first round messages sent to any pair of honest servers pass the pairwise verification checks. Since, $|I_1 \cup I_2 \cup K_0| \leq 4\lambda = t$, the reduction emulates an adversary that corrupts at most t servers. Thus, if the messages generated by the external challenger are done as in the real execution then input to the distinguisher is identical to Hyb_3. Else, it is identically distributed to Hyb_4. This implies that the reduction breaks the security of the outer protocol and this is a contradiction. □

7 Black-Box Protocol Compilers in the Multiparty Setting

We state our main theorems about our protocol compiler in the multiparty case. The proof of these theorems are given in the Appendix.

7.1 Protocol Compiler in the Random Oracle Model

In this subsection, we give a construction of a three-round malicious-secure MPC protocol in the random oracle model that makes black-box use of a two-round semi-honest OT. It was shown in [ABG+20] that even considering only semi-honest security in the random oracle model, such a black-box protocol for the case of three parties is round-optimal. Recently, [PS21] gave a malicious-secure construction in the CRS model assuming a two-round malicious secure oblivious transfer protocol that additionally satisfies equivocal receiver security [GS18].

We give the formal statement of our theorem below.

Theorem 7.1. *Let f be an arbitrary n-party functionality. Assuming the existence of:*

- *A two-round, 2-client, m-server pairwise verifiable MPC protocol $\Phi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ for computing f against t server corruptions (see Definition 5.2).*
- *A two-round semi-honest oblivious transfer protocol $\mathsf{OT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{out}_{\mathsf{OT}})$.*

Then, there exits a three-round protocol Γ for computing f over point-to-point channels that makes black-box use OT and satisfies security with selective abort against static, malicious adversaries in the random oracle model. The communication and computation costs of the protocol are $\mathsf{poly}(\lambda, n, |f|)$, where $|f|$ denotes the size of the circuit computing f.

Instantiating the pairwise verifiable MPC protocol from Theorem 5.4, we get the following corollary.

Corollary 7.2. *Let f be an arbitrary n-party functionality. There exits a three-round protocol Γ for computing f over point-to-point channels that makes black-box use OT and satisfies security with selective abort against static, malicious adversaries in the random oracle model. The communication and computation costs of the protocol are $\mathsf{poly}(\lambda, n, |f|)$, where $|f|$ denotes the size of the circuit computing f.*

For simplicity, we give our construction over broadcast channels and note that we can use similar techniques as in [IKSS21] to transform this protocol to the point-to-point channels.

In Section 7.1.1, we describe the construction of the above malicious-secure protocol over broadcast channels and in Section 7.1.2, we give the proof of security.

7.1.1 Construction

Building Blocks. The construction makes use of the following building blocks.

1. A two-round n-client, m-sever protocol $\Phi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ that is pairwise verifiable protocol w.r.t. predicate P for computing f against t server corruptions. We set $t = 5\lambda n^3$ and $m = 6t + 1$.

2. A three-round inner protocol $\Pi_j = (\Pi_{j,1}, \Pi_{j,2}, \Pi_{j,3}, \mathsf{out}_\Pi)$ for each $j \in [m]$ where Π_j computes the function $\mathsf{Eval}(j, \cdot)$ (i.e., the function computed by the j-th server). For each $j \in [m]$, we require protocol Π_j to satisfy the following properties:

   (a) It has publicly decodable transcript [ABG+20]. This means that out_Π only takes the transcript of the protocol as input and computes the output (without making use of any secret information).

   (b) Π_j is a parallel composition of α sub-protocols where each sub-protocol is computing a special functionality called $\mathsf{3MULTPlus}$ [BGI+18, GIS18, ABG+20].[8] The sub-protocol has publicly decodable transcript and the number of parties involved in each of the sub-protocols is constant.

[8]The $\mathsf{3MULTPlus}$ functionality is a three-party functionality where the i-th party's input for $i \in [3]$ is given by $(x_i, y_i) \in \{0,1\} \times \{0,1\}$. The functionality outputs $x_1 \cdot x_2 \cdot x_3 \oplus y_1 \oplus y_2 \oplus y_3$. It was shown in [BGI+18, GIS18, ABG+20] that a protocol for $\mathsf{3MULTPlus}$ functionality that has publicly decodable transcript can be bootstrapped to a protocol for arbitrary functions making black-box use of a PRG.

(c) The sub-protocol satisfies the standard correctness and the following weak-adaptive semi-honest security definition: for any adversary $\mathcal{A}$ corrupting a subset of the parties M (and let H denote the set of uncorrupted parties) in the sub-protocol, there exists a stateful simulator Sim such that for every input of the honest parties $\{x_i\}_{i\in H}$, we have:

$$\{\mathsf{Real}(1^\lambda, \mathcal{A}, \{x_i\}_{i\in H})\}_\lambda \approx_c \{\mathsf{Ideal}(1^\lambda, \mathcal{A}, \mathsf{Sim}, \{x_i\}_{i\in H})\}_\lambda$$

where the experiments Real and Ideal are described in Figure 4.

(d) Π_j also satisfies weak-adaptive semi-honest security property and this is proved via an hybrid argument where we change each of the α sub-protocols to the Ideal experiment described in Figure 4.

We provide a construction of such a sub-protocol for computing the $\mathsf{3MULTPlus}$ functionality in Appendix B and the construction of Π_j (specifically, property (2d)) follows via standard completeness results of $\mathsf{3MULTPlus}$ functionality [BGI+18, GIS18, ABG+20].

$\mathsf{Real}(1^\lambda, \mathcal{A}, \{x_i\}_{i\in H})$

(a) Sample uniform random tape r_i for each $i \in [n]$.

(b) Send $\{r_i\}_{i\in M}$ to $\mathcal{A}$.

(c) Generate the first round message from the honest parties using input $\{x_i\}_{i\in H}$ and the random tape $\{r_i\}_{i\in H}$ and send them to $\mathcal{A}$.

(d) $\mathcal{A}$ sends the first round message along with the input $\{x_i\}_{i\in M}$.

(e) For the second and third rounds,

i. Generate the messages from the honest parties using the input $\{x_i\}_{i\in H}$ and the random tape $\{r_i\}_{i\in H}$ and send it to $\mathcal{A}$.

ii. Receive the messages for this round from $\mathcal{A}$ on behalf of the malicious parties.

(f) Compute the output of the protocol using the public decoder.

(g) Output the view of $\mathcal{A}$ and the output of the honest parties.

$\mathsf{Ideal}(1^\lambda, \mathcal{A}, \mathsf{Sim}, \{x_i\}_{i\in H})$

(a) Sample uniform random tape r_i for each $i \in M$.

(b) Send $\{r_i\}_{i\in M}$ to $\mathcal{A}$.

(c) Generate the first round message from the honest parties in the protocol using $\mathsf{Sim}(1^\lambda)$ and send it to $\mathcal{A}$.

(d) Receive the first round message from $\mathcal{A}$ along with the inputs $\{x_i\}_{i\in M}$.

(e) For the second and third rounds,

i. Provide Sim with $\{x_i, r_i\}_{i\in M}$.

ii. If the messages received in the previous rounds from the adversarial parties are inconsistent with the random tape $\{r_i\}_{i\in M}$ and input $\{x_i\}_{i\in M}$ or if $\mathcal{A}$ issues a corrupt command, then provide the honest parties inputs $\{x_i\}_{i\in H}$ to Sim.

iii. If it is the final round message, then provide Sim with output of the functionality being computed.

iv. Generate the messages from the honest parties in this round using Sim and send them to $\mathcal{A}$.

v. Receive the messages from $\mathcal{A}$ on behalf of the malicious parties.

(f) Compute the output of the protocol using the public decoder

(g) Output the view of $\mathcal{A}$ and the output of the honest parties.

Figure 4: Description of the Real and Ideal experiments.

- **Round-1:** Each party P_i does the following:
  1. It computes $(x_1^i, \ldots, x_m^i, vk_i) \leftarrow \mathsf{Share}(1^\lambda, i, x_i)$.
  2. It sets $\mathbb{T} = \phi$ and for each $j \in [m]$, sets $\mathbb{T}_j = \phi$. Here, $\mathbb{T}$ denotes the transcript of the entire protocol seen so far and $\mathbb{T}_j$ refers to the messages corresponding to Π_j in the transcript $\mathbb{T}$.
  3. For each $j \in [m]$,
     (a) It computes $r_j^i := H_1(i, j, x_j^i, s_j^i)$ for uniformly chosen $s_j^i \leftarrow \{0,1\}^\lambda$.
     (b) It computes $\mathsf{com}_j^i \leftarrow \mathsf{Com}((x_j^i, s_j^i))$.
     (c) It computes $\pi_{j,1}^i \leftarrow \Pi_{j,1}(i, x_j^i, \mathbb{T}_j; r_j^i)$.
  4. It computes $K_1^i = H_2(i, \mathbb{T} \| \{\mathsf{com}_j^i, \pi_{j,1}^i\}_{j\in[m]}, \mathsf{tag}_1^i)$ where $\mathsf{tag}_1^i \leftarrow \{0,1\}^\lambda$.
  5. It broadcasts $\{\mathsf{com}_j^i, \pi_{j,1}^i\}_{j\in[m]}$, tag_1^i, and $\{(x_j^i, s_j^i), \mathsf{Open}(\mathsf{com}_j^i)\}_{j\in K_1^i}$.
  6. At the end of the round, P_i appends the messages sent and received in this round to $\mathbb{T}$ and the corresponding messages sent and received in the protocol Π_j to $\mathbb{T}_j$ for each $j \in [m]$.
- **Rounds-**2&3**:** In round $u \in [2,3]$, each P_i does the following:
  1. It runs $\mathsf{checkConstMPC}(u, \mathbb{T})$ where $\mathsf{checkConstMPC}$ is described in Figure 6. If $\mathsf{checkConstMPC}$ outputs 0, then it aborts.
  2. Else, for each $j \in [m]$,
     (a) It computes $\pi_{j,u}^i \leftarrow \Pi_{j,u}(i, x_j^i, \mathbb{T}_j; r_j^i)$.
  3. It computes $K_u^i = H_2(i, \mathbb{T} \| \{\pi_{j,u}^i\}_{j\in[m]}, \mathsf{tag}_u^i)$ where $\mathsf{tag}_u^i \leftarrow \{0,1\}^\lambda$.
  4. It broadcasts $\{\pi_{j,u}^i\}_{j\in[m]}$, tag_u^i, and $\{(x_j^i, s_j^i), \mathsf{Open}(\mathsf{com}_j^i)\}_{j\in K_u^i}$.
  5. At the end of the round, P_i appends the messages sent and received in this round to $\mathbb{T}$ and the corresponding messages sent and received in the protocol Π_j to $\mathbb{T}_j$ for each $j \in [m]$.
- **Output:** To compute the output, P_i does the following:
  1. It runs $\mathsf{checkConstMPC}(4, \mathbb{T})$ where $\mathsf{chkConsistency}$ is described in Figure 6. If $\mathsf{checkConstMPC}$ outputs 0, then it aborts and outputs $\perp$.
  2. For each $j \in [m]$,
     (a) It runs $\mathsf{out}_{\Pi_j}(\mathbb{T}_j)$ to obtain ϕ_j.
  3. It runs Dec on $i, vk_i, (\phi_1, \ldots, \phi_m)$ and outputs whatever it outputs.

Figure 5: Description of the Black-Box Three-Round MPC Protocol

3. A non-interactive, straight-line extractable commitment $(\mathsf{Com}, \mathsf{Open})$. Such a commitment scheme can be constructed unconditionally in the random oracle model [Pas03] (refer Section 4.3).

4. Two hash functions $H_1 : \{0,1\}^* \to \{0,1\}^*$ and $H_2 : \{0,1\}^* \to (\{0,1\}^k)^m$ that are modelled as random oracles where k is the number of random bits needed to toss a biased coin that outputs 1 with probability $\lambda n^2/2m$. We interpret the output of H_2 as a subset K of $[m]$ where each element of $[m]$ is included in K independently with probability $\lambda n^2/2m$.

Description of the Protocol. We give the formal description of the construction in Figure 5.

Input: A round number $j \in [2,4]$ and the transcript $\mathbb{T}$.

1. For each $i \in [n]$:
   (a) It computes the sets $\{K_1^i, \ldots, K_{j-1}^i\}_{i \in [n]}$ from the transcript $\mathbb{T}$ and the hash function H_2.
   (b) Let K^i be the union of the sets $K_1^i, \ldots, K_{j-1}^i$.
   (c) For each $j \in K^i$,
      i. It obtains $\{(x_j^i, s_j^i), \mathsf{Open}(\mathsf{com}_j^i)\}$ from $\mathbb{T}$.
      ii. It checks if $\mathsf{Open}(\mathsf{com}_j^i)$ is valid.
      iii. It then checks if $(x_j^i, H(i,j,x_j^i,s_j^i))$ is a valid (input,randomness) pair for the protocol Π_j consistent with the transcript $\mathbb{T}_j$.
      iv. For each $j' \in K^i$, it checks if $P(i,j,j',x_j^i,x_{j'}^i) = 1$.
2. If any of the checks fail, it outputs 0. Else, if all the checks pass, it outputs 1.

Figure 6: Description of checkConstMPC

7.1.2 Proof of Security

Let $\mathcal{A}$ be the malicious adversary that is corrupting a subset M of the parties. Let H be the set of honest parties. We start with the description of the simulator Sim.

Description of Sim.

1. **Interaction with the Environment.** For every input value corresponding to the corrupted parties that Sim receives from the environment, it writes these values to the input tape of the adversary $\mathcal{A}$. Similarly, the contents of the output tape of $\mathcal{A}$ is written to Sim's output tape. To simulate the interaction with $\mathcal{A}$, Sim does the following.

2. For each round $u \in [3]$, Sim chooses uniform subsets K_u^i of $[m]$ where each element is included with probability $\lambda n^2/2m$ for each $i \in H$ and it sets I_2 to be the union of all these sets. If for any $i \in H$ and $u \in [3]$, if $|K_u^i| \geq \lambda n^2$, then Sim aborts. Note that $|I_2| \leq 3\lambda n^3$. It programs the random oracle H_2 to output the appropriate set when queried on the messages generated by P_i.

3. Sim starts interacting with the simulator Sim_Φ for the outer protocol by corrupting the set of clients indexed by M and the set of servers indexed by I_2. It obtains the first round messages $\{x_j^i\}_{j \in I_2}$ sent by the honest client to the corrupted servers.

4. **Round-1 Message from Sim.** For each $j \in I_2$, it uses the messages $\{x_j^i\}_{j \in I_2}$ and uniformly chosen $\{s_j^i\}_{j \in I_2}$ to generate the messages in the protocol Π_j as described in Figure 1. For each $j \notin I_2$, it runs the simulator for the inner protocol Π_j to generate the first round messages on behalf of P_i for each $i \in H$. To generate the first round commitments for each $j \in I_2$, Sim uses (x_j^i, s_j^i) to honestly generate com_j^i. For each $j \notin I_2$, Sim generates a commitment to dummy values.

5. **Round-1 Message from $\mathcal{A}$.**

(a) For each of the unique random oracle queries made by $\mathcal{A}$, Sim samples a uniform element in the range of the oracle and outputs it as the response. Each time Sim generates query to the random oracle on behalf of honest P_i, Sim checks if adversary has already made that query. If that is the case, then it aborts the execution and outputs a special symbol ABORT.

(b) On obtaining the first round message from $\mathcal{A}$, Sim uses the straight-line extractor for the extractable commitment Com and obtains $\{(x_1^i, s_1^i), \ldots, (x_m^i, s_m^i)\}_{i \in M}$ from $\{\mathsf{com}_1^i, \ldots, \mathsf{com}_m^i\}_{i \in M}$.

6. **Checks done by** Sim**.**

(a) Sim initializes an empty set I_1.

(b) For each $j \notin I_2$, Sim provides $\{(x_j^i, H_1(i, j, x_j^i, s_j^i))\}_{i \in M}$ as the input and the randomness pair used by the adversarial parties to the corresponding simulator for Π_j.

(c) In every round, for each $j \in [m]$, it checks if for each $i \in M$ that $(x_j^i, H(i, j, x_j^i, s_j^i))$ is a valid (input,randomness) pair for the protocol Π_j w.r.t. the $\mathbb{T}_j$ or if the PRG computations done in x_j^i are correct. If not, it adds j to the set I_1. It adaptively corrupts the server j in the outer protocol and obtains $\{x_j^i\}_{i \in H}$. It then adaptively corrupts the honest clients in the inner protocol Π_j by providing $\{x_j^i\}_{i \in H}$ as the honest party inputs to Sim_{Π_j}. It uses the messages received from this simulator on behalf of the honest parties $\{P_i\}_{i \in H}$ to execute the rest of the protocol Π_j.

(d) For each $i \in M$, it constructs an inconsistency graph G_i on vertices from $[m]$ where it adds an edge between j, j' if $P(i, j, j', x_j^i, x_{j'}^i) = 0$. It then computes a 2-approximation for the min-vertex cover on this graph and sets this to be I_3^i. It sets $I_3 = \cup_{i \in M} I_3^i$. For each $j \in I_3$, it adaptively corrupts the server j in the outer protocol and obtains $\{x_j^i\}_{i \in H}$. It then adaptively corrupts the honest clients in the inner protocol Π_j by providing $\{x_j^i\}_{i \in H}$ as the honest party inputs to Sim_{Π_j}. It uses the messages received from this simulator on behalf of the honest parties $\{P_i\}_{i \in H}$ to execute the rest of the protocol Π_j.

(e) If $|I_1| \geq \lambda n^3$ or $|I_3| \geq \lambda n^3$, then it sends $\perp$ to its ideal functionality.

(f) At the end of each round if $\mathcal{A}$'s messages does not pass $\mathsf{checkConstMPC}$ then Sim sends $\perp$ to the trusted functionality.

7. **Second Round Message from** Sim **and** $\mathcal{A}$**.**

(a) Sim sends the second round message for the executions $j \notin I_1 \cup I_2 \cup I_3$ using the simulator for Π_j. For each $j \in I_1 \cup I_2 \cup I_3$, Sim generates the messages on behalf of the honest parties as explained before.

8. **Second Round Message from** $\mathcal{A}$ **to** Sim**.**

(a) Sim receives the second round messages from $\mathcal{A}$.

(b) Sim updates the set I_1 as and performs the same checks as before.

9. **Final Round Message from** Sim**.**

(a) Before sending the final round message, Sim provides $\{x_j^i\}_{i \in M, j \notin I_1 \cup I_2 \cup I_3}$ to Sim_Φ as the messages sent by the adversary to the honest servers. Sim_Φ queries the ideal functionality on an input $\{x_i\}_{i \in M}$ and Sim forwards this to its trusted functionality.

(b) Sim obtains $f(x_1, \ldots, x_n)$ from the ideal functionality and forwards this to Sim_Φ. Sim_Φ sends the second round protocol messages $\{\phi_j\}_{j \notin I_1 \cup I_2 \cup I_3}$ from the honest servers. For each $j \notin I_1 \cup I_2 \cup I_3$, Sim uses ϕ_j as the output of Π_j and runs the simulator for Π_j on this output to generate the final round message.

10. **Output Computation.** To compute the output, Sim updates the set I_1 and performs the same checks as described before. It adaptively corrupts the newly added servers to I_1. If Sim_Φ instructs the honest party P_i for some $i \in H$ to output $\perp$, then Sim forwards this instruction to this honest party. Otherwise, it instructs the ideal functionality to output $f(x_1, \ldots, x_n)$ to the other uncorrupted parties.

Proof of Indistinguishability. We now argue that the real execution and the ideal execution are computationally indistinguishable via a hybrid argument.

- $\underline{\mathsf{Real}}$: This corresponds to the output of the real execution of the protocol.
- $\underline{\mathsf{Hyb}_0}$: This hybrid corresponds to the distribution where the adversaries random oracle queries are answered with an uniformly chosen random element from the image of the oracle. Further, if the adversary makes any queries to the hash functions H_1, H_2 before the exact same query was made by the honest party, we abort. We note that since each query made to the hash functions H_1, H_2 has a component which is uniformly chosen random string of length λ, the probability that an adversary is able to make a query that exactly matches this string queried by an honest party is $q \cdot 2^{-\lambda}$ where q is the number of queries that $\mathcal{A}$ makes to the random oracle. Hence, this hybrid is statistically close to the previous one.
- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we make the following changes:

  1. We use the straight-line extractor for the extractable commitment Com to obtain $\{(x_1^i, s_1^i), \ldots, (x_m^i, s_m^i)\}_{i \in M}$ from $\{\mathsf{com}_1^i, \ldots, \mathsf{com}_m^i\}_{i \in M}$.
  2. We construct the set I_1 and I_3 as described in the simulation.
  3. If $|I_1| \geq \lambda n^3$ or $|I_3| \geq \lambda n^3$, we abort the execution and instruct all the honest parties to abort the execution.

  In Lemma 7.3, we show that Hyb_0 and Hyb_1 are statistically indistinguishable.
- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we make the following changes:

  1. For each round $u \in [3]$, we choose a subset K_u^i of $[m]$ for each $i \in H$ where each element in $[m]$ is independently included with probability $\lambda n^2 / 2m$. If for any $i \in H$, $u \in [3]$, if $|K_u^i| \geq \lambda n^2$, we abort. We set I_2 to be the union of all these sets. We program the random oracle H_2 to output the appropriate set when queried on the messages generated by P_i.
  2. For each $j \notin I_2$, we change the commitments com_j^i to be commitments to some dummy values instead of (x_j^i, s_j^i).

  Note that via standard Chernoff bound, the probability that for any $i \in H$ and $u \in [3]$, the probability that $|K_u^i| \geq \lambda n^2$ is $2^{-O(\lambda n)}$. Therefore, the probability that the simulation aborts is negligible. Further, we observe that this hybrid is computationally indistinguishable to the previous hybrid from the hiding property of the non-interactive commitment scheme.

- $\underline{\mathsf{Hyb}_3}$: In this hybrid, we make the following changes:
  1. For each round $u \in [3]$ and for each $i \in H$, we choose uniform subsets K_u^i (of size $\lambda \cdot n$) and set I_2 to be the union of all these sets. We program the random oracle H_2 to output the appropriate set when queried on the messages generated by P_i.
  2. For each $j \notin I_2$, we run the simulator for the inner protocol and generate the messages from P_i for each $i \in H$ for the protocol Π_j using this simulator.
  3. We send $\{x_j^i, H_1(i, j, x_j^i, s_j^i)\}_{i \in M, j \notin I_2}$ to the simulator for Π_j.
  4. If some $j \notin I_2$ is added to I_1 or I_3 before it sends its final round message in the protocol, we send the honest party inputs $\{x_j^i\}_{i \in H}$ to the simulator. We use the messages received from the simulator on behalf of the honest parties to complete the rest of the execution Π_j.
  5. If $|I_1| \geq \lambda n^3$ in any round or if $|I_3| \geq \lambda n^3$, then we abort as in the previous hybrid.
  6. For $j \notin I_1 \cup I_2$, we use the input $\{x_j^i\}_{i \in M}$ extracted from the extractable commitment to compute ϕ_j as $\mathsf{Eval}(j, x_j^1, \ldots, x_j^n)$.
  7. For each $j \notin I_1 \cup I_2 \cup I_3$, we send this output ϕ_j to the simulator for Π_j and obtain the final round message in Π_j. We use this to generate the final round message in the overall protocol.
  8. We then compute the output of the protocol as in the previous hybrid.

  In Lemma 7.4, we show that $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$ from the weak-adaptive semi-honest security of the inner protocol and the specific properties it satisfies (see the paragraph Building Blocks in Section 7.1.1).
- $\underline{\mathsf{Hyb}_4}$: In this hybrid, we make the following changes:
  1. We (adaptively) corrupt the set of servers corresponding to the indices $I_1 \cup I_2 \cup I_3$ and the clients $\{P_i\}_{i \in M}$. We run the simulator Sim_Φ for the outer protocol and obtain the first round messages sent by the honest clients to these corrupted servers. We use this to complete the execution with $\mathcal{A}$.
  2. We provide $\{x_j^i\}_{i \in M, j \notin I_1 \cup I_2 \cup I_3}$ (extracted from the extractable commitment) to Sim_Φ as the messages sent by the adversary to the honest servers. Sim_Φ queries the ideal functionality on an input $\{x_i\}_{i \in M}$.
  3. We compute $z = f(x_1, \ldots, x_n)$ and send this to Sim_Φ as the output from the ideal functionality. Sim_Φ sends the second round protocol messages $\{\phi_j\}_{j \notin I_1 \cup I_2 \cup I_3}$ from the honest servers. We use this to generate the final round message of the protocol as in the previous hybrid.
  4. To compute the output, if Sim_Φ instructs the honest party P_i for some $i \in H$ to output $\perp$, then Sim forwards this instruction to this honest party. It then instructs the other honest parties to output z.

  In Lemma 7.5, we show that $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$ from the security of the outer protocol. We note that output of Hyb_4 is identically distributed to the output of the ideal execution with Sim.

Lemma 7.3. $\mathsf{Hyb}_0 \approx_s \mathsf{Hyb}_1$

Proof. We show that if $|I_1| \geq \lambda n^3$ or $|I_3| \geq \lambda n^3$, then each honest client in Hyb_0 also aborts with overwhelming probability.

1. **If** $|I_1| \geq \lambda n^3$**:** Let u be the first round when $|I_1| \geq \lambda n^2$. By a standard averaging argument, there exists at least one party $i \in M$, that has cheated in over $\lambda \cdot n^2$ executions. Let I_1^i be the set of executions where this party has cheated. Note that K_u^i is chosen as the response of the random oracle after the adversary generates the u-th round message in the protocol on behalf of the party P_i. We show that since K_u^i is sampled such that each element in $[m]$ is independently included with probability $\lambda n^2/2m$, the probability that $|I_1^i \cap K_u^i| = 0$ is $2^{-O(\lambda)}$. Note that if this event doesn't happen, then every honest client aborts in Hyb_0.

$$\Pr[|K_u^i \cap I_1^i| = 0] \quad \leq \quad \left(1 - \frac{\lambda \cdot n^2}{2m}\right)^{\lambda \cdot n^2} < e^{-O(\lambda)} \,.$$

   where the last inequality follows since $m = O(\lambda n^3)$. By an union bound over the set of all the q queries that adversary makes to the random oracle H_2, the probability that there exists some K_u^i which is a response of the RO such that $|K_u^i \cap I_1^i| = 0$ is upper bounded by $q \cdot e^{-O(\lambda)}$.

2. **If** $|I_3| \geq \lambda n^3$**:** By a standard averaging argument, there exists at least one $i \in M$ such that $|I_3^i| \geq \lambda n^2$. Since $|I_3^i| \geq \lambda n^2$, the size of the minimum vertex cover is at least $\lambda n^2/2$. This means that in the inconsistency graph there exists a maximum matching of size at least $\lambda n^2/4$ edges. Let M be the set of vertices for this matching. If there exists at least one edge of this matching in K_u^i then all the honest parties abort in Hyb_0. Fix some edge in the matching. The probability this edge is not included in K_u^i is at most $(1-(\lambda n^2/2m)^2) = 1-O(1/n^2)$. The probability that no edge of the matching is present in K_u^i is at most $(1 - O(1/n^2))^{\lambda n^2/4} = e^{-O(\lambda)}$. By an union bound over the set of all the q queries that adversary makes to the random oracle H_2, the probability that there exists some K_u^i which is a response of the RO such that $|K_u^i \cap I_3^i| = 0$ is upper bounded by $q \cdot e^{-O(\lambda)}$.

□

Lemma 7.4. *Assuming the weak-adaptive semi-honest security of the inner protocol and the specific properties it satisfies (see Building Blocks in Section 7.1.1), we have that* $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$.

Proof. For every round u and for every P_i such that $i \in H$, we sample a uniform set K_u^i of size $\lambda \cdot n$ and we fix I_2 to be the union of all these sets. We program the random oracle H_2 to output the appropriate set when queried on the messages generated by P_i.

Let $I = [m] \setminus I_2$. Let α be the number of executions of the $\mathsf{3MULTPlus}$ protocol in the inner protocol Π_j for each $j \in [m]$. We consider a sequence of $\alpha|I|$ hybrids between Hyb_2 and Hyb_3 (see Property (2d)) where we change from real to simulated executions of each of the α executions of the $\mathsf{3MULTPlus}$ protocol for each $j \in I$ one by one. If Hyb_2 and Hyb_3 are computationally distinguishable, then by a standard hybrid argument, there exists two sub-hybrids Hyb_2' and Hyb_3' which differ only in one execution of $\mathsf{3MULTPlus}$ protocol. Specifically, in Hyb_3', the messages in the β-th execution (for some $\beta \in [\alpha]$) of $\mathsf{3MULTPlus}$ protocol in Π_j is generated as in the ideal execution and in the Hyb_2' it is generated as in the real execution. We now show that this contradicts the weak-adaptive semi-honest security of the sub-protocol.

Let us denote the subset of honest parties that interact in the β-th execution of $\mathsf{3MULTPlus}$ in Π_j as $H_{j,\beta}$ and the set of corrupted parties as $M_{j,\beta}$. We note that $M_{j,\beta}$ is a constant sized set (see Property (2b)). We begin interacting with external challenger and provide $\{x^i_{j,\beta}\}_{i\in H_{j,\beta}}$ as the input used by the honest clients in the particular execution of the $\mathsf{3MULTPlus}$ protocol. The challenger provides with a random tape $\{r^i_{j,\beta}\}_{i\in M_{j,\beta}}$ to be used by the adversarial parties. Amongst all the queries made by $\mathcal{A}$ to the random oracle H_1 where the first two inputs are (i,j) for each $i \in M_{j,\beta}$, we choose one of these queries (i,j,x^i_j,s^i_j) at random and we embed $r^i_{j,\beta}$ as the randomness to be used in the β-th execution of $\mathsf{3MULTPlus}$ in the response from the random oracle. On receiving the first round message from $\mathcal{A}$, we run the extractor for the extractable commitment Com on $\{\mathsf{com}^i_j\}_{i\in M_{j,\beta}}$ and obtain $\{(\overline{x}^i_j,\overline{s}^i_j)\}_{i\in M_{j,\beta}}$. We extract the input $\overline{x}^i_{j,\beta}$ that is used in the β-th execution of $\mathsf{3MULTPlus}$ in Π_j. We provide $\{\overline{x}^i_{j,\beta}\}_{i\in M_{j,\beta}}$ as the input used by the corrupt clients to the external challenger along with the appropriate first round messages in this sub-protocol. We consider the following cases.

1. We check if j is added to I_1 or I_3 at the end of the first round. If that is the case, then we issue the corrupt command to the challenger. We continue with the rest of the execution by using the messages sent by the challenger and compute the output as before. We finally run the distinguisher between Hyb'_2 and Hyb'_3 on the view of the adversary and the output of the honest parties and output whatever the distinguisher outputs.

2. If j is not added to I_1 or I_3 at the end of the first round, but $(\overline{x}^i_j,\overline{s}^i_j) \neq (x^i_j,s^i_j)$ for some $i \in M_{j,\beta}$, then we output a random bit to the external challenger.

3. If j is not added to I_1 or I_3 at the end of the first round and $(\overline{x}^i_j,\overline{s}^i_j) = (x^i_j,s^i_j)$ for every $i \in M_{j,\beta}$, then we continue the interaction with the external challenger and use these messages to generate the messages in the β-th $\mathsf{3MULTPlus}$ execution in Π_j. We compute the output of the protocol as before. We finally run the distinguisher between Hyb'_2 and Hyb'_3 on the view of the adversary and the output of the honest parties and output whatever the distinguisher outputs.

We note that in Cases-1 and 3, the input to the distinguisher is identically distributed to Hyb'_2 if the messages generated by the challenger are as per the real execution and otherwise, it is identically distributed to Hyb'_3. Finally, conditioning on j not added to I_1 or I_3 after the first round, the probability that there exists an $i \in M_{j,\beta}$ such that $(\overline{x}^i_j,\overline{s}^i_j) = (x^i_j,s^i_j)$ is at least $1-(1/q)^{|M_{j,\beta}|}-\mathsf{negl}(\lambda)$ (and is bounded above by $1-(1/q)^{|M_{j,\beta}|}+\mathsf{negl}(\lambda)$) where q is the total number of queries made by the adversary to the random oracle H_1. Let us assume that the probability that the distinguisher correctly predicts whether it is given a sample from Hyb'_2 and Hyb'_3 to be $1/2+\mu(\lambda)$ (for some non-negligible $\mu(\lambda)$). Let ϵ be the probability that j is added to I_1 or I_3 at the end of the first round. Let p be the probability that the above reduction correctly predicts whether it is interacting with the real execution or the ideal execution. Then,

$$\begin{aligned}
p &\geq (1/2+\mu(\lambda))\epsilon + (1-\epsilon)((1-(1/q)^{|M_{j,\beta}|}-\mathsf{negl}(\lambda))(1/2) + ((1/q)^{|M_{j,\beta}|}-\mathsf{negl}(\lambda))(1/2+\mu(\lambda))) \\
&\geq (1/2+\mu(\lambda))\epsilon + (1-\epsilon)(1/2+\mu(\lambda)/q^{|M_{j,\beta}|}) - \mathsf{negl}(\lambda) \\
&\geq 1/2+\mu(\lambda)/q^{|M_{j,\beta}|} + \epsilon(\mu(\lambda)-\mu(\lambda)/q^{|M_{j,\beta}|}) - \mathsf{negl}(\lambda) \\
&\geq 1/2+\mu(\lambda)/q^{|M_{j,\beta}|} - \mathsf{negl}(\lambda)
\end{aligned}$$

and this contradicts the weak-adaptive semi-honest security of the inner protocol as $|M_{j,\beta}|$ is a constant.

□

Lemma 7.5. *Assuming the security of the outer protocol* Φ*, we have* $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$.

Proof. Assume for the sake of contradiction that Hyb_3 and Hyb_4 are computationally distinguishable. We give a reduction to breaking the security of the outer protocol.

We begin interacting with the external challenger by providing the inputs $\{x_i\}_{i \in H}$ as the honest client's inputs. We then corrupt the clients indexed by M and the set of servers indexed by I_2. We obtain the first round messages sent from the honest client P_i to the corrupted servers and we begin interacting with $\mathcal{A}$ using these messages. For each server that is added to I_1 or I_3, we adaptively corrupt that server and obtain the first round message sent from the honest clients to this server. We use these messages to adaptively corrupt the clients $\{P_i\}_{i \in H}$ of the inner protocol. We use the messages generated by the simulator to continue with the rest of the execution. Before sending the final round message on behalf of the honest clients, we send $\{x^i_j\}_{i \in M, j \notin I_1 \cup I_2 \cup I_3}$ as the first round messages sent by the corrupted clients to the honest servers.

The challenger replies with $\{\phi_j\}_{j \notin I_1 \cup I_2 \cup I_3}$ as the final round message from the honest servers sent to the corrupt client. We use this as the input to the inner protocol simulator to generate the final round messages from $\{P_i\}_{i \in H}$. On receiving the final round message from the adversary, we update the set I_1 and adaptively corrupt the newly added servers to I_1. The challenger provides the output of $\{P_i\}_{i \in H}$ and we instruct the parties to output the same. We finally run the distinguisher between Hyb_3 and Hyb_4 on the view of $\mathcal{A}$ and the output of $\{P_i\}_{i \in H}$ and output whatever the distinguisher outputs.

Note that $|I_2| \leq 3\lambda n^3$, $|I_1| \leq \lambda n^3$ and $|I_3| \leq \lambda n^3$. Further, the reduction emulates an admissible adversary. Hence, $|I_1 \cup I_2 \cup I_3| \leq t$. Thus, the reduction emulates an adversary that corrupts at most t servers. Thus, if the messages generated by the external challenger are done as in the real execution then input to the distinguisher is identical to Hyb_3. Else, it is identically distributed to Hyb_4. This implies that the reduction breaks the security of the outer protocol and this is a contradiction. □

7.2 Protocol Compiler in the OT Correlations Model

In this subsection, we improve the result from [IKSS21] and give a construction of a two-round black-box protocol for computing multiparty functionalities with security against malicious adversaries in the OT correlations model. This compiler makes black-box use of a two-round semi-malicious secure inner protocol that has first message equivocality (defined in [IKSS21] and recalled in Definition 7.6).

Building Blocks. The construction makes use of the following building blocks.

1. A two-round n-client, m-sever protocol $\Phi = (\Phi_1, \Phi_2, \mathsf{out}_\Phi)$ satisfying privacy with knowledge of outputs[9] for computing the function $g((x_1, k_1), \ldots, (x_n, k_n)) = (y = f(x_1, \ldots, x_n), \{\mathsf{MAC}(k_i, y)\}_{i \in [n]})$ where MAC is a strongly unforgeable one-time MAC scheme. This protocol is secure against t server corruptions and has publicly decodable transcript. We set $t = (m-1)/3$ and $m = 16\lambda n^3$. Such a protocol was constructed in [IKP10, Pas12] by making black-box use

[9] Privacy with knowledge of outputs is a weaker notion than security with selective abort and allows the adversary to select the output given by the trusted functionality to the honest parties. We refer the reader to [IKP10] for the formal definition.

of a PRG. 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.

2. A two-round inner protocol $\Pi_j = (\Pi_{j,1}, \Pi_{j,2}, \mathsf{out}_\Pi)$ with publicly decodable transcript for each $j \in [m]$ where Π_j computes the function $\Phi_2(j, \cdot)$ (i.e., the function computed by the j-th server). For each $j \in [m]$, we require protocol Π_j to satisfy the following definition.

   Definition 7.6 ([IKSS21])**.** *We say that $(\Pi_1, \Pi_2, \mathsf{out}_\Pi)$ is a two-round, inner protocol for computing a function f with publicly decodable transcript if it satisfies the following properties:*

   - ***Correctness:*** *We say that the protocol Π correctly computes a function f if for every choice of inputs x_i for party P_i and for any choice of random tape r_i, we require that for every $i \in [n]$,*

   $$\Pr[\mathsf{out}_\Pi(i, \pi(2)) = f(x_1, \ldots, x_n)] = 1$$

     where $\pi(2)$ denotes the transcript of the protocol Π when the input of P_i is x_i with random tape r_i and sk_i is the output key generated by Π_1.
   - ***Security.*** *Let $\mathcal{A}$ be an adversary corrupting a subset of the parties indexed by the set M and let H be the set of indices denoting the honest parties. We require the existence of a simulator Sim_Π such that for any choice of honest parties inputs $\{x_i\}_{i \in H}$, we have:*

   $$\textit{Real}(\mathcal{A}, \{x_i, r_i\}_{i \in H}) \approx_c \textit{Ideal}(\mathcal{A}, \mathsf{Sim}_\Pi, \{x_i\}_{i \in H})$$

     where the real and ideal experiments are described in Figure 7 and for each $i \in H$, r_i is uniformly chosen.

   [IKSS21] showed that the protocol from [GIS18] in the OT correlations model and [LLW20] in the OLE correlations model satisfy the above definition.

3. A single round Rabin OT protocol $\mathsf{RabinOT}$ with erasure probability $1 - \lambda \cdot n/m$. We extend the syntax of the Rabin OT protocol to take in m strings and each of these strings are independently erased with probability $1 - \lambda \cdot n/m$.

Theorem 7.7. *Let f be an arbitrary n-party functionality. Assume the existence of:*

- *A two-round n-client, m-sever protocol $\Phi = (\Phi_1, \Phi_2, \mathsf{out}_\Phi)$ satisfying privacy with knowledge of outputs against t server corruptions for computing the function g defined above.*
- *A two-round inner protocol $\Pi_j = (\Pi_{j,1}, \Pi_{j,2}, \mathsf{out}_\Pi)$ with publicly decodable transcript for each $j \in [m]$ where Π_j computes the function $\Phi_2(j, \cdot)$ (i.e., the function computed by the j-th server) satisfying Definition 7.6.*

Then, there exists a two-round protocol Γ that makes black box use of $\{\Pi_j\}_{j \in [m]}$ and computes f against static, malicious adversaries satisfying security with selective abort in the 1-out-of-2 OT correlations model and access to point-to-point channels. Further, if only $(\Phi_1, \mathsf{out}_\Phi)$ makes black-box use of a PRF and Φ_2 does not perform any cryptographic operations, then Γ is fully black-box. The communication and computation costs of the protocol are $\mathsf{poly}(\lambda, n, |f|)$, where $|f|$ denotes the size of the circuit computing f and the size of the OT correlations shared between the parties is a fixed polynomial in the security parameter and number of parties and is independent of the size of the function f.

As in the previous section, we give the description of protocol over broadcast channels and we can use the same techniques outlined in [IKSS21] to transform it to the point-to-point channel.

$\mathsf{Real}(\mathcal{A}, \{x_i, r_i\}_{i \in H})$	$\mathsf{Ideal}(\mathcal{A}, \mathsf{Sim}_\Pi, \{x_i\}_{i \in H})$
(a) For each $i \in H$, compute $\pi_1^i := \Pi_1(1^\lambda, i, x_i; r_i)$.	(a) For each $i \in H$, compute $\pi_1^i := \mathsf{Sim}_\Pi(1^\lambda, i)$.
(b) Send $\{\pi_1^i\}_{i \in H}$ to $\mathcal{A}$.	(b) Send $\{\pi_1^i\}_{i \in H}$ to $\mathcal{A}$.
(c) Receive $\{\pi_1^i, (x_i, r_i)\}_{i \in M}$ from $\mathcal{A}$.	(c) Receive $\{\pi_1^i, (x_i, r_i)\}_{i \in M}$ from $\mathcal{A}$.
(d) Check if the messages sent by corrupt parties in $\pi(1)$ are consistent with $\{x_i, r_i\}_{i \in M}$.	(d) Check if the messages sent by corrupt parties in $\pi(1)$ are consistent with $\{x_i, r_i\}_{i \in M}$.
(e) **Semi-Malicious Security:** If they are consistent:	(e) **Semi-Malicious Security:** If they are consistent:
i. For each $i \in H$, compute $\pi_2^i := \Pi_2(1^\lambda, i, x_i, \pi(1); r_i)$.	i. For each $i \in H$, compute $\pi_2^i \leftarrow \mathsf{Sim}_\Pi(1^\lambda, i, f(x_1, \ldots, x_n), \{x_j, r_j\}_{j \in M}, \pi(1))$.
(f) **Equivocality:** If they are not consistent:	(f) **Equivocality:** If they are not consistent:
i. For each $i \in H$, compute $\pi_2^i := \Pi_2(1^\lambda, i, x_i, \pi(1); r_i)$.	i. For each $i \in H$, compute $\pi_2^i \leftarrow \mathsf{Sim}_\Pi(1^\lambda, i, \{x_i\}_{i \in H}, \pi(1))$.
(g) Send $\{\pi_2^i\}_{i \in H}$ to $\mathcal{A}$.	(g) Send $\{\pi_2^i\}_{i \in H}$ to $\mathcal{A}$.
(h) Receive $\{\pi_2^i\}_{i \in M}$ from $\mathcal{A}$.	(h) Receive $\{\pi_2^i\}_{i \in M}$ from $\mathcal{A}$.
(i) Output the view of $\mathcal{A}$ and $\{\mathsf{out}_\Pi(i, \pi(2))\}_{i \in H}$.	(i) Output the view of $\mathcal{A}$ and $\{\mathsf{out}_\Pi(i, \pi(2))\}_{i \in H}$.

Figure 7: Security Game for the Two-Round Inner Protocol

Description of the Protocol. We give the formal description of the protocol in Figure 8.

7.2.1 Proof of Security

Let $\mathcal{A}$ be an adversary that corrupts the set of parties indexed by M and let $H := [n] \setminus M$.

Description of Sim. The simulator Sim is given below:

1. **Interaction with Environment.** For every input value corresponding to the corrupted parties that Sim receives from the environment, it writes these values to the input tape of the adversary $\mathcal{A}$. Similarly, the contents of the output tape of $\mathcal{A}$ is written to Sim's output tape. To simulate the interaction with $\mathcal{A}$, Sim does the following.

2. **Rabin-OT Setup:**
   - For every $i \in M$ and $j \in H$, Sim samples a subset $K_{i,j}$ of $[m]$ where each element is added to $K_{i,j}$ independently with probability $\lambda \cdot n/m$. If for any $j \in H$, $|K_{i,j}| \geq 2\lambda n$, it aborts.

3. Let $C = \cup_{i \in M, j \in H} K_{i,j}$. Sim invokes Sim_Φ by corrupting the set of clients indexed by M and corrupting the set of servers indexed by C. Sim_Φ provides $\{x_j^i\}_{i \in H, j \in C}$.

4. For each $h \in C$, Sim chooses an uniform random tape $\{r_h^i\}_{i \in H}$. For every $i \in H$ and $h \in C$, Sim uses x_h^i as the input and r_h^i as the random tape of P_i to generate the first round message in the protocol Π_h.

- **Round-1:** In the first round, the party P_i with input x_i does the following:
  1. It chooses a random MAC key $k_i \leftarrow \{0,1\}^*$ and sets $z_i := (x_i, k_i)$.
  2. It computes $(x_1^i, \ldots, x_m^i) \leftarrow \Phi_1(1^\lambda, i, z_i)$.
  3. It chooses a random string $r_h^i \leftarrow \{0,1\}^*$ for every $h \in [m]$ and sets $y_h^i = \{r_h^i, x_h^i\}$.
  4. For each $j \in [n] \setminus \{i\}$, it computes $\mathsf{msg}^{j,i} \leftarrow \mathsf{RabinOT}(y_1^i, \ldots, y_m^i)$.
  5. For each $h \in [m]$, it computes $\pi_{h,1}^i := \Pi_{h,1}(1^\lambda, i, x_h^i; r_h^i)$.
  6. It broadcasts $\{\pi_{h,1}^i\}_{h \in [m]}, \{\mathsf{msg}^{j,i}\}_{j \in [n] \setminus \{i\}}$.
- **Round-2:** In the second round, P_i does the following:
  1. It decrypts $\{\mathsf{msg}^{i,j}\}_{j \in [n] \setminus \{i\}}$ to obtain $\{r_h^j, x_h^j\}_{j \in [n] \setminus \{i\}, h \in K_{i,j}}$ for some set $K_{i,j}$ (the other positions are erased).
  2. For each $j \in [n] \setminus \{i\}$ and $h \in K_{i,j}$, it checks:
     (a) If the PRG computations in x_h^j are correct.
     (b) If $\pi_{h,1}^j := \Pi_{h,1}(1^\lambda, j, x_h^j; r_h^j)$.
  3. If any of the above checks fail, it aborts.
  4. Else, for each $h \in [m]$, it computes $\pi_{h,2}^i := \Pi_{h,2}(1^\lambda, i, x_h^i, \pi_h(1); r_h^i)$ (where $\pi_h(1)$ denotes the transcript in the first round of Π_h).
  5. It broadcasts $\{\pi_{h,2}^i\}_{h \in [m]}$ to every party.
- **Output Computation.** To compute the output, P_i does the following:
  1. If any party has aborted, then abort.
  2. Else, for every $h \in [m]$, it computes $\phi_h := \mathsf{out}_{\Pi_h}(i, \pi_h(2))$ (where $\pi_h(2)$ denotes the transcript in the first two rounds of Π_h).
  3. It runs out_Φ on $\{\phi_h\}_{h \in [m]}$ to obtain $(y, \sigma_1, \ldots, \sigma_m)$.
  4. It checks if σ_i is a valid tag on y using the key k_i. If yes, it outputs y and otherwise, it aborts.

Figure 8: Description of the Two-Round Black-Box Malicious MPC protocol in the OT Correlations Model

5. For each $i \in H$ and $j \in M$, Sim computes $\mathsf{msg}^{j,i}$ by setting $\{y_h^i\}_{h \notin C}$ to be junk values. For each $h \in C$, it sets $y_h^i = (r_h^i, x_h^i)$. For each $i \in H$ and $j \in H$, Sim computes $\mathsf{msg}^{j,i}$ using junk values as inputs.
6. For each $h \notin C$, Sim invokes Sim_{Π_h} to generate the first round message $\{\pi_h^i\}_{i \in H}$. It sends $\{\pi_{h,1}^i, \{\mathsf{ct}_h^{j,i}\}_{j \in [n] \setminus \{i\}}\}_{h \in [m]}$ on behalf of each $i \in H$. It receives the first round message from $\mathcal{A}$.
7. It uses the Rabin-OT extractor to decode $\{\mathsf{msg}^{j,i}\}_{j \in H, i \in M}$.
8. For each $h \in [m]$, Sim checks if for every $i \in M$ there exists some $j \in H$, y_h^i (obtained from $\mathsf{msg}^{j,i}$) contains the input and randomness that explains the messages sent by corrupt parties in Π_h as well as contains the correct PRG computations. If not, it adds h to a set C' (which is initially empty). If such a j exists, then for every $i \in M$, Sim uses (x_h^i, r_h^i) present in y_h^i as the consistent input and randomness used by corrupt party P_i in the protocol Π_h.
9. If $|C'| > \lambda n^3$, then Sim instructs the ideal functionality to send abort to all the honest parties and outputs the view of the adversary.

10. If $|C'| \leq \lambda n^3$, then Sim instructs Sim_Φ to adaptively corrupt the servers indexed by C' and obtains $\{x_h^i\}_{i\in H, h\in C'}$. For each $i \in H$ and $j \in M$, Sim chooses a random subset $K_{i,j}$ where each element of $[m]$ is independently added with probability $\lambda \cdot n/m$. If for any $i \in H$ and $h \in K_{i,j}$, $\{y_h^j\}_{j\in M}$ (derived from $\mathsf{msg}^{i,j}$) contains inconsistent input and randomness or if the PRG computations are incorrect, then Sim instructs the ideal functionality to send abort to i. Let H' be the subset of honest parties that have not aborted.

11. For every $h \in [m] \setminus \{C \cup C'\}$, Sim sends $\{x_h^i\}_{i\in M}$ to Sim_Φ. Sim_Φ queries the ideal functionality $\{(x_i, k_i)\}_{i\in M}$ and Sim forwards $\{x_i\}_{i\in M}$ to its own ideal functionality. It It obtains y from the trusted functionality. For each $i \in M$, it computes $\sigma_i := \mathsf{MAC}(k_i, y)$ and for each $i \in H$, it chooses σ_i uniformly at random. It forwards $(y, \sigma_1, \ldots, \sigma_n)$ as the response to Sim_Φ. Sim_Φ replies with $\{\phi_h\}_{h\in[m]\setminus\{C\cup C'\}}$.

12. To generate the final round message,
    - For each $h \in [m] \setminus \{C \cup C'\}$, Sim sends $\{(x_h^i, r_h^i)\}_{i\in M}$ as the input and the randomness of corrupt parties and ϕ_h as the output of the function computed by Π_h to Sim_{Π_h}. Sim_{Π_h} generates the last round message on behalf of the parties in H' in Π_h.
    - For each $h \in C'$, Sim sends $\{x_h^i\}_{i\in H}$ as the inputs of the honest parties to Sim_{Π_h} and obtains the last round message on behalf of H'.
    - For each $h \in C$, Sim uses $\{r_h^i, x_h^i\}_{i\in H}$ to generate the final round message on behalf of $H' \subseteq H$.

13. To compute the output for each P_i where $i \in H$
    - If $H' \neq H$, then Sim instructs the ideal functionality to output abort to all the honest parties.
    - For each $h \in C' \cup C$, Sim derives ϕ_h using out_{Π_h}.
    - It then computes $(y', \sigma'_1, \ldots, \sigma'_n) := \mathsf{out}_\Phi(\{\phi_h\}_{h\in[m]})$.
    - Sim checks if $y' = y$ and for each $i \in H$, that $\sigma'_i = \sigma_i$. For every $i \in H$, such that above check passes, Sim instructs the ideal functionality to deliver the outputs to P_i. For all other parties, Sim instructs them to abort.

Proof of Indistinguishability. We now argue that the real and the ideal executions are computationally indistinguishable by a hybrid argument.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the view of the adversary and the outputs of the honest parties in the real execution of the protocol.
- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we make the following changes to the first round message generated by the honest parties. Specifically, for every $i \in H$,
  - If $j \in M$ and $h \notin C$, we set y_h^i used in generating in $\mathsf{msg}^{j,i}$ as junk values.
  - If $j \in H$, then for each $h \in [m]$, we set y_h^i used in generating $\mathsf{msg}^{j,i}$ as junk values.

  The computational indistinguishability between Hyb_0 and Hyb_1 follows immediately from the security of the Rabin OT protocol.
- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we define the set C as in the simulation. For every $h \notin C$, we generate the protocol messages in Π_h using the simulator Sim_{Π_h}. Specifically,
  - For every $h \notin C$, to generate the first round message on behalf of the honest parties, we run Sim_{Π_h} and obtain $\{\pi_h^i\}_{i\in H}$.
  - We then use the Rabin OT extractor to extract from $\{\mathsf{msg}^{j,i}\}_{j\in H, i\in M}$.

- For each $h \in [m]$, we check if for every $i \in M$, there exists some $j \in H$ such that y_h^i (derived from $\mathsf{msg}^{j,i}$) contains the input and randomness that explains the messages sent by corrupt parties in Π_h as well as contains the correct PRG computations. If not, we add h to a set C' (which is initially empty). If such a j exists, then for every $i \in M$, we use (x_h^i, r_h^i) present in y_h^i as the consistent input and randomness used by corrupt party P_i in the protocol Π_h.
- For each $i \in H$ and $j \in M$, we choose a random subset $K_{i,j}$ of $[m]$ where each element is added independently to $[m]$ with probability $\lambda \cdot n/m$. If for any $h \in K'_{i,j}$, y_h^j (derived from $\mathsf{ct}_h^{i,j}$) contains inconsistent input and randomness or if the PRG computations are incorrect, then we instruct the honest P_i to abort. Let H' be the subset of honest parties that have not aborted.
- For each $h \in [m] \setminus \{C \cup C'\}$, we send $\{(x_h^i, r_h^i\}_{i \in M}$ as the input and the randomness of corrupt parties and ϕ_h (computed honestly using $\{x_h^i\}_{i \in H}$) as the output of the function computed by Π_h to Sim_{Π_h}. Sim_{Π_h} generates the last round message on behalf of the parties in H' in Π_h.
- For each $h \in C'$, we send $\{x_h^i\}_{i \in H}$ as the inputs of the honest parties to Sim_{Π_h} and obtain the last round message on behalf of H'.
- For each $h \in C$, we uses $\{r_h^i, x_h^i\}_{i \in H}$ to generate the final round message on behalf of $H' \subseteq H$.
- To compute the output, we do the same steps as described in the protocol.

We show in Lemma 7.8 that Hyb_2 is computationally indistinguishable to Hyb_1 from the security of the inner protocol.

- $\underline{\mathsf{Hyb}_3}$: In this hybrid, we define the set C' as in the simulation and if $|C'| > \lambda n^3$ at the end of the first or the second round, then we abort. Also, if for any $i \in M$ and $j \in H$, the sampled $K_{i,j}$ is such that $|K_{i,j}| \geq 2\lambda n$, it aborts.
  We show in Lemma 7.9 that $\mathsf{Hyb}_2 \approx_s \mathsf{Hyb}_3$.
- $\underline{\mathsf{Hyb}_4}$: In this hybrid, we use the simulator Sim_Φ to generate the protocol messages for the outer protocol, instead of running honest party strategy.
  We argue in Claim 7.10 that Hyb_3 is computationally indistinguishable to Hyb_4.
  $\underline{\mathsf{Hyb}_5}$: In this hybrid, we make the following two changes:
  - When Sim_Φ queries the ideal functionality g on $\{x_i, k_i\}_{i \in M}$, we query f on $\{x_i\}_{i \in M}$ and obtain the output y. For each $i \in M$, we compute $\sigma_i := \mathsf{MAC}(k_i, y)$ and for each $i \in H$, we choose σ_i uniformly at random.
  - In the output phase, we recover $(y', \sigma'_1, \ldots, \sigma'_n)$ as in the previous hybrid and then check if $y' = y$ and if for each $i \in H$, if $\sigma'_i = \sigma_i$. For every $i \in H$, such that above check passes, we instruct the ideal functionality to deliver the outputs to P_i. For all other parties, we instruct them to abort.

  This hybrid is statistically close to Hyb_4 from the strong unforgeability of one-time MACs and the uniformity of Tags under a randomly chosen key. We note that Hyb_5 is identically distributed to the ideal world using Sim.

Lemma 7.8. *Assuming the security of the inner MPC protocol, we have* $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$.

Proof. Most parts of this proof are taken verbatim from [IKSS21]. Assume for the sake of contradiction that there exists a distinguisher D that can distinguish Hyb_1 from Hyb_2 with non-negligible advantage. By a standard averaging argument, this implies that there exists $h \in [m] \setminus C$ and two distributions (described below) $\mathsf{Hyb}_{1,h}$ and $\mathsf{Hyb}_{1,h-1}$ (where $\mathsf{Hyb}_{1,0} \equiv \mathsf{Hyb}_1$) such that D can distinguish between $\mathsf{Hyb}_{1,h}$ and $\mathsf{Hyb}_{1,h-1}$ with non-negligible advantage. In both these distributions, for every $k < h$ such that $k \in [m] \setminus C$, the messages in the protocol Π_k are generated using the simulator Sim_{Π_k} and for every $k > h$ and $k \in [m] \setminus C$, the messages in the protocol Π_k are generated using the real algorithms. The only difference between these two distributions is how the messages in protocol Π_h are generated. In $\mathsf{Hyb}_{1,h}$, they are generated using Sim_{Π_h} and in $\mathsf{Hyb}_{1,h-1}$, they are generated using the real algorithms. Note that $\mathsf{Hyb}_{1,|[m]\setminus C|}$ is distributed identically to Hyb_2. We now construct an adversary $\mathcal{B}$ that uses D and breaks security of the inner protocol.

$\mathcal{B}$ interacts with the external challenger and sends $\{x_h^i\}_{i \in H}$ as the inputs of the honest parties. It obtains the first round message $\{\pi_{h,1}^i\}$ from the external challenger and it generates the rest of the components in the first round message on behalf of each honest party as in $\mathsf{Hyb}_{1,h-1}$. It sends the first round message on behalf of each honest party to $\mathcal{A}$ and receives the first round message sent by $\mathcal{A}$ on behalf of the malicious parties. $\mathcal{B}$ then uses the sampled keys $\{k_h^{j,i}\}_{j \in H, i \in M, h \in [m]}$ in the OT correlations setup phase to decrypt the ciphertexts $\{ct_h^{j,i}\}_{j \in H, i \in M, h \in [m]}$ received from $\mathcal{A}$.

$\mathcal{B}$ checks if for every $i \in M$ there exists some $j \in H$ such that y_h^i (derived from $\mathsf{ct}_h^{j,i}$) contains the input and randomness that explains the messages sent by corrupt parties in Π_h as well as contains the correct PRG computations. If yes, for every $i \in M$, $\mathcal{B}$ uses (x_h^i, r_h^i) present in y_h^i as the consistent input and randomness used by corrupt party P_i in the protocol Π_h. It sends this to the external challenger. Otherwise, $\mathcal{B}$ sends some dummy input and the randomness on behalf of each malicious party to the external challenger. $\mathcal{B}$ receives the final round message $\{\pi_{h,2}^i\}_{i \in H}$ and uses this to generate the final round message of the overall protocol exactly as in $\mathsf{Hyb}_{1,h-1}$. To compute the output, $\mathcal{B}$ performs the same steps as in the protocol. Finally, $\mathcal{B}$ runs D on the view of the adversary and the outputs of the honest parties and outputs whatever D outputs.

Note that if the protocol messages in Π_h were generated by the external challenger using the real algorithms, then the input to D is distributed identically to $\mathsf{Hyb}_{1,h-1}$. Otherwise, it is distributed identically to $\mathsf{Hyb}_{1,h}$. Since D can distinguish $\mathsf{Hyb}_{1,h-1}$ and $\mathsf{Hyb}_{1,h}$ with non-negligible advantage, $\mathcal{B}$ can break the security of the inner protocol which is a contradiction. $\square$

Lemma 7.9. $\mathsf{Hyb}_2 \approx_s \mathsf{Hyb}_3$.

Proof. Note that for any $i \in M$ and $j \in H$, the probability that sampled $|K_{i,j}| \geq 2\lambda n$ is $2^{-O(\lambda)}$ (by standard Chernoff bounds). Thus, by standard union bound, the probability that there exists an $i \in M$ and $j \in H$ such that $|K_{i,j}| \geq 2\lambda n$ is $2^{-O(\lambda)}$.

If $|C'| > \lambda n^3$ then by a standard averaging argument, there exists some $j \in M$ that cheats in more than λn^2 executions. Let us call these executions as C'_j. Fix some honest party $i \in H$. We show that probability that $|K_{i,j} \cap C'_j| = 0$ is $2^{-O(\lambda)}$.

Since each element in C'_j is added independently to $K_{i,j}$ with probability $\lambda \cdot n/m$, the probability that no element of C'_j is added to $K_{i,j}$ is at most $(1 - \frac{\lambda \cdot n}{m})^{\lambda n^2} \leq 2^{-O(\lambda)}$ (since $m = 16\lambda n^3$).

The above argument proves that party $i \in H$ aborts with probability at least $1 - 2^{-O(\lambda)}$ in the case where $|C'| > \lambda n^3$. To complete the proof of the claim, we observe via a union bound that the probability that there exists at least one honest party that does not abort is at most $n \cdot 2^{-O(\lambda)}$. $\square$

Lemma 7.10. *Assuming the security of the outer MPC protocol, we have* $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$.

Proof. Most parts of this proof are taken verbatim from [IKSS21]. Assume for the sake of contradiction that there exists a distinguisher D that can distinguish between Hyb_3 and Hyb_4 with non-negligible advantage. We now use D to construct an adversary $\mathcal{B}$ that can break the security of the outer MPC protocol.

$\mathcal{B}$ $\{x_i\}_{i \in H}$ as the inputs of the honest clients in the outer MPC protocol to the external challenger.

During the Rabin-OT setup phase $\mathcal{B}$ uses the sampled $\{K_{i,j}\}_{i \in M, j \in H}$ and sets $C := \{K_{i,j}\}_{i \in M, j \in H}$. $\mathcal{B}$ instructs the external challenger to corrupt the set of clients given by M and the set of servers given by C.

The challenger provides $\{x_j^i\}_{i \in H, j \in C}$. $\mathcal{B}$ uses this to generate the messages in the protocol $\{\Pi_h\}_{h \in C}$. At the end of the first round, $\mathcal{B}$ constructs the set C' as in Hyb_3. If $|C'| \leq \lambda n^3$, then $\mathcal{B}$ instructs the external challenger to corrupt the servers indexed by C' and obtains $\{x_h^i\}_{i \in H, h \in C'}$. For every $h \in [m] \setminus \{C \cup C'\}$, $\mathcal{B}$ sends $\{x_h^i\}_{i \in M}$ as the first round messages generated by corrupted client to the honest server indexed by h. $\mathcal{B}$ obtains $\{\phi_h\}_{h \in [m] \setminus \{C \cup C'\}}$. $\mathcal{B}$ uses this to generate the final round message of the protocol as in Hyb_3.

On receiving the final round message from $\mathcal{A}$, $\mathcal{B}$ computes $\{\phi_h\}_{h \in [m]}$ using the public decoder for Π_h and runs out_Φ on $(\phi_1, \ldots, \phi_m)$ to obtain $(y, \sigma_1, \ldots, \sigma_m)$. It then performs the same MAC checks as in Hyb_3 to compute the output. $\mathcal{B}$ runs D on the view of the adversary and the outputs of the honest parties and outputs whatever D outputs.

Since $|C| < 2\lambda n^3$ and $|C'| \leq \lambda n^3$, the size of $|C \cup C'| < 3\lambda n^3 < (m-1)/3$. Note that if the messages of the outer protocol are generated by the real algorithms, then the inputs to D are distributed identically to Hyb_3. Else, they are identically distributed to Hyb_4. Thus, if D can distinguish between Hyb_3 and Hyb_4 with non-negligible advantage then $\mathcal{B}$ breaks the security of the outer MPC protocol, which is a contradiction. □

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.

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *45th FOCS*, pages 166–175, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press.

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[BCG+19a] 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.

[BCG+19b] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. LNCS, pages 489–518, Santa Barbara, CA, USA, 2019. Springer, Heidelberg, Germany.

[BCR86] Gilles Brassard, Claude Crépeau, and Jean-Marc Robert. Information theoretic reductions among disclosure problems. In *27th FOCS*, pages 168–173, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press.

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

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In Gary L. Miller, editor, *STOC 96*, pages 479–488. ACM, 1996.

[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.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In Janos Simon, editor, *STOC 1988*, pages 1–10, 1988.

[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.

[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.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

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

[FMV19] Daniele Friolo, Daniel Masny, and Daniele Venturi. A black-box construction of fully-simulatable, round-optimal oblivious transfer from strongly uniform key agreement. In Dennis Hofheinz and Alon Rosen, editors, *TCC*, volume 11891 of *Lecture Notes in Computer Science*, pages 111–130. Springer, 2019.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194, Santa Barbara, CA, USA, August 1987. 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.

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

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

[HIK+11] Iftach Haitner, Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions of protocols for secure computation. *SIAM J. Comput.*, 40(2):225–266, 2011.

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

[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 *EUROCRYPT*, 2011.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30, San Diego, CA, USA, June 11–13, 2007. ACM Press.

[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*, 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.

[IPS09] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In *TCC 2009*, pages 294–314, 2009.

[LLW20] Huijia Lin, Tianren Liu, and Hoeteck Wee. Information-theoretic 2-round MPC without round collapsing: Adaptive security, and more. In *TCC 2020*, volume 12551 of *Lecture Notes in Computer Science*, pages 502–531. Springer, 2020.

[MR17] Payman Mohassel and Mike Rosulek. Non-interactive secure 2pc in the offline/online and batch settings. In *Eurocrypt 2017*, volume 10212 of *Lecture Notes in Computer Science*, pages 425–455, 2017.

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

[MRR20] Ian McQuoid, Mike Rosulek, and Lawrence Roy. Minimal symmetric PAKE and 1-out-of-n OT from programmable-once public functions. In *CCS 2020*, pages 425–442. ACM, 2020.

[NP99] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *31st ACM STOC*, pages 245–254, Atlanta, GA, USA, May 1–4, 1999. ACM Press.

[ORS15] Rafail Ostrovsky, Silas Richelson, and Alessandra Scafuro. Round-optimal black-box two-party computation. In *CRYPTO 2015, Part II*, pages 339–358, 2015.

[Pas03] Rafael Pass. On deniability in the common reference string and random oracle model. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 316–337, Santa Barbara, CA, USA, August 17–21, 2003. 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.

[Yao86] 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.

A Semi-Honest Oblivious Transfer with Weak Adaptive Security

In this section, we give a construction of a semi-honest secure oblivious transfer (OT) that satisfies certain restricted forms of adaptive security (which we call as *weak adaptive security*). We give the formal description of the security properties in Section A.1. In Section A.2, we give a construction that satisfies this definition based on any semi-honest oblivious transfer. In Section A.3, we give the proof of security.

A.1 Definition

Syntax. Consider two parties, a sender with input $(m_0, m_1) \in \{0,1\}^* \times \{0,1\}^*$ (where $|m_0| = |m_1|$) and a receiver with an input bit b. A r-round oblivious transfer protocol between a sender and a receiver is given by a set of algorithms $(\mathsf{OT}_1, \ldots, \mathsf{OT}_r)$ and the output decoder $\mathsf{out}_{\mathsf{OT}}$. Here, OT_i denotes the algorithm that computes the message to be sent in the i-th round of the protocol. This algorithm takes as input the security parameter 1^λ, the transcript in the first $(i-1)$ rounds, the input and the internal randomness of the party speaking in the i-th round and outputs the i-th round message to be sent by this party. When $i = (r-1)$, this algorithm additionally outputs a secret key sk. $\mathsf{out}_{\mathsf{OT}}$ is run by the receiver and takes as input the r-round transcript of the protocol and the secret key sk and generates the output of the receiver.

We give the security definition below.

Security Definition. We give the formal security definition below.

Definition A.1. *An r-round oblivious transfer protocol $(\mathsf{OT}_1, \ldots, \mathsf{OT}_r, \mathsf{out}_{\mathsf{OT}})$ between a sender and a receiver is said to satisfy weak adaptive semi-honest security if the following properties hold:*

- ***Correctness:*** *For every input $b \in \{0,1\}$ of the receiver and the inputs $(m_0, m_1) \in \{0,1\}^*$ of the sender (where $|m_0| = |m_1|$), we have:*

$$\Pr[\mathsf{out}_{\mathsf{OT}}((\mathsf{msg}_1, \ldots, \mathsf{msg}_r), sk) = m_b] = 1$$

  where $(\mathsf{msg}_1, \mathsf{msg}_1, \ldots, \mathsf{msg}_r)$ denotes the r-round transcript generated by the algorithms $(\mathsf{OT}_1, \ldots, \mathsf{OT}_r)$ using inputs b of the receiver and (m_0, m_1) of the sender and sk is the secret key output by OT_{r-1}.
- ***Weak Adaptive Semi-Honest Sender Security.*** *There exists a (stateful) simulator Sim_S such that for every (stateful) adversary $\mathcal{A}$ corrupting the receiver and any sender inputs (m_0, m_1) (such that $|m_0| = |m_1|$), we have:*

$$\{\mathsf{Real}_S(1^\lambda, \mathcal{A}, (m_0, m_1))\}_\lambda \approx_c \{\mathsf{Ideal}_S(1^\lambda, \mathcal{A}, \mathsf{Sim}_S, (m_0, m_1))\}_\lambda$$

  where the distributions Real_S and Ideal_S are described in Figure 9.
- ***Weak Adaptive Semi-Honest Receiver Security.*** *There exists a (stateful) simulator Sim_R such that for every (stateful) adversary $\mathcal{A}$ corrupting the sender and any receiver input b, we have:*

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

  where the distributions Real_R and Ideal_R are described in Figure 10.

$\mathsf{Real}_S(1^\lambda, \mathcal{A}, (m_0, m_1))$	$\mathsf{Ideal}_S(1^\lambda, \mathcal{A}, \mathsf{Sim}_S, (m_0, m_1))$
1. Sample uniform random tapes r for the receiver and s for sender.	1. Sample a uniform random tape of the receiver r.
2. Send r to $\mathcal{A}$.	2. Send r to $\mathcal{A}$.
3. $\mathcal{A}$ outputs the receiver input b.	3. $\mathcal{A}$ outputs the receiver input b.
4. For each $i \in [r-1]$,	4. For each $i \in [r-1]$,
(a) Generate the message msg_i to be sent in the i-th round of the protocol using the input, the random tape sampled above and the previous round messages.	(a) If the i-th message is sent by the receiver, then use the random tape r and the input b to generate msg_i. If the i-th message is sent by the sender, then use the simulator Sim_S to generate the i-th round message msg_i.
(b) Run $\mathcal{A}(\mathsf{msg}_1, \ldots, \mathsf{msg}_i)$.	(b) Run $\mathcal{A}(\mathsf{msg}_1, \ldots, \mathsf{msg}_i)$.
(c) If $\mathcal{A}$ outputs a special symbol $\mathsf{corrupt}$, then send the random tape of the sender to $\mathcal{A}$ and output the view of $\mathcal{A}$. Otherwise, increment i.	(c) If $\mathcal{A}$ outputs a special symbol $\mathsf{corrupt}$, then run the simulator on (m_0, m_1) to obtain the random tape of the sender. Send this to $\mathcal{A}$ and output its view. Otherwise, increment i.
5. Generate the last round message msg_r from sender and send this to $\mathcal{A}$.	5. Generate the last round message msg_r using $\mathsf{Sim}_S(b, r, m_b)$ and send this to $\mathcal{A}$.
6. Output the view of $\mathcal{A}$.	6. Output the view of $\mathcal{A}$.

Figure 9: Descriptions of Real_S and Ideal_S.

A.2 Construction

In this subsection, we give a black-box transformation from any r-round, semi-honest secure OT protocol to a r-round OT protocol satisfying Definition A.1.

A.2.1 Description of the Protocol

We give the description of the protocol below.

Construction. We run two instances of the semi-honest OT protocol on random inputs c_0, c_1 for the receiver and (s_0^0, s_0^1) and (s_1^0, s_1^1) for the sender respectively. In the pre-final round, the receiver additionally sends $d_0 = b \oplus c_0$ and $d_1 = b \oplus c_1$ to the sender. The secret key output by OT_{r-1} corresponds to one of the randomly chosen executions $\beta \in \{0,1\}$, c_β, the secret key of the β-th instance of the semi-honest protocol along with b. In the final round, the sender additionally sends $(x_0^0 = m_0 \oplus s_0^{d_0}, x_0^1 = m_1 \oplus s_0^{1 \oplus d_0})$ and $(x_1^0 = m_0 \oplus s_1^{d_1}, x_1^1 = m_1 \oplus s_1^{1 \oplus d_1})$. To retrieve the output, we use the secret key of the β-th semi-honest OT protocol to obtain $s_\beta^{c_\beta}$ and then, retrieve m_b from x_β^b.

$\mathsf{Real}_R(1^\lambda, \mathcal{A}, b)$	$\mathsf{Ideal}_R(1^\lambda, \mathcal{A}, \mathsf{Sim}_R, b)$
1. Sample uniform random tape r for the receiver and s for sender.	1. Sample uniform random tape s for the sender.
2. Send s to $\mathcal{A}$.	2. Send s to $\mathcal{A}$.
3. $\mathcal{A}$ outputs (m_0, m_1) where $\|m_0\| = \|m_1\|$.	3. $\mathcal{A}$ outputs (m_0, m_1) where $\|m_0\| = \|m_1\|$.
4. For each $i \in [r-2]$,	4. For each $i \in [r-2]$,
(a) Generate the message msg_i to be sent in the i-th round of the protocol using the input, the random tape sampled above and the previous round messages.	(a) If the i-th message is sent by the sender, then use s and the inputs (m_0, m_1) to generate the i-th round message msg_i. If the i-th message is sent by the receiver, then use the simulator Sim_R to generate the i-th round message msg_i.
(b) Run $\mathcal{A}(\mathsf{msg}_1, \ldots, \mathsf{msg}_i)$.	(b) Run $\mathcal{A}(\mathsf{msg}_1, \ldots, \mathsf{msg}_i)$.
(c) If $\mathcal{A}$ outputs a special symbol $\mathsf{corrupt}$, then send the random tape of the receiver to $\mathcal{A}$ and output view of $\mathcal{A}$. Otherwise, increment i.	(c) If $\mathcal{A}$ outputs a special symbol $\mathsf{corrupt}$, then run the simulator on b to obtain the random tape of the receiver. Send this and to $\mathcal{A}$ and output its view. Otherwise, increment i.
5. Generate the pre-final round message msg_{r-1} from the receiver and send this to $\mathcal{A}$.	5. Generate the pre-final round message msg_{r-1} using Sim_R and send this to $\mathcal{A}$.
6. If the adversary outputs a special symbol $\mathsf{corrupt}$, then send sk (which is output by OT_{r-1}) to $\mathcal{A}$.	6. If the adversary outputs a special symbol $\mathsf{corrupt}$, then run Sim_R on b to obtain sk and send this to $\mathcal{A}$.
7. Output the view of $\mathcal{A}$.	7. Output the view of $\mathcal{A}$.

Figure 10: Descriptions of Real_R and Ideal_R.

A.3 Proof of Security

Correctness follows directly from the correctness of the semi-honest secure OT protocol and the information-theoretic reduction from random OT to specific input OT.

A.3.1 Weak Adaptive Semi-Honest Sender Security.

We start with the description of Sim_S.

Description of Sim_S. Sim_S chooses two pairs of random strings (s_0^0, s_0^1) and (s_1^0, s_1^1). It generates all the sender messages in the first $(r-1)$ rounds of the protocol honestly using the above sampled strings as the inputs. If the adversary issues a $\mathsf{corrupt}$ command in any of these rounds, then it reveals the random tape used to generate the sender messages. To generate the final round message, it obtains (b, r, m_b) where r is the random tape of the receiver. It obtains the choice bits c_0, c_1 that the adversary used in the two OT executions from the random tape r. It sets $x_0^b = m_b \oplus s_0^{c_0}$ and $x_1^b = m_b \oplus s_1^{c_1}$. It then chooses x_0^{1-b} and x_1^{1-b} uniformly at random. It generates the final round message of the two instances of the semi-honest OT protocol honestly and sends this to the adversary along with (x_0^0, x_0^1) and (x_1^0, x_1^1).

Proof of Indistinguishability. We show that the output of Real_S is computationally indistinguishable to the output of Ideal_S via a hybrid argument.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the output of the experiment Real_S.
- $\underline{\mathsf{Hyb}_1}$: In this hybrid, if the adversary does not issue a $\mathsf{corrupt}$ command, then in the final round message, we sample x_0^{1-b} and x_1^{1-b} uniformly at random.
  In Lemma A.2, we show that Hyb_0 and Hyb_1 are computationally indistinguishable.
  We note that Hyb_1 is identical to Ideal_S.

Lemma A.2. *Assuming the semi-honest sender security of the OT protocol, 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. We give a reduction that breaks the semi-honest sender security of the underlying OT protocol.

The reduction interacts with an external challenger that generates the messages for two instances of the semi-honest OT protocol. The reduction chooses random bits c_0, c_1 uniformly and a random string $s_0^{c_0}, s_1^{c_1}$ uniformly. It provides c_0, c_1 as the challenge receiver inputs and $s_0^{c_0}, s_1^{c_1}$ as the challenge receiver outputs to the external challenger. It chooses two pairs of random strings $(s_0^{1-c_0}, \overline{s}_0^{1-c_0})$ and $(s_1^{1-c_1}, \overline{s}_1^{1-c_1})$ uniformly and gives them as the challenge sender input strings to the external challenger. It obtains the random tapes r_0, r_1 of the receiver from the external challenger for the two OT executions. It sends (r_0, c_0, r_1, c_1) as the random tape to the adversary. The adversary outputs the receiver input b. The reduction then chooses a uniform bit χ (as the guess of whether the adversary will issue the $\mathsf{corrupt}$ command or not). If $\chi = 0$ (denoting the guess that adversary issues the $\mathsf{corrupt}$ command), then it generates the messages to be sent to the adversary honestly using (s_0^0, s_0^1) and (s_1^0, s_1^1) as the input strings in the two OT executions respectively. If $\chi = 1$, then it forwards the messages received from the external challenger in both the OT executions to the adversary. Before sending the final round message, if the adversary issues a $\mathsf{corrupt}$ command but $\chi = 1$, or if adversary does not issue the $\mathsf{corrupt}$ command but $\chi = 0$, then the reduction outputs a random bit to the external challenger and aborts the interaction with the adversary. On the other hand, if adversary issues a $\mathsf{corrupt}$ command and $\chi = 0$, then it provides the adversary with random tape (which includes the strings (s_0^0, s_0^1) and (s_1^0, s_1^1)) used to generate the messages in the protocol. If adversary does not issue the $\mathsf{corrupt}$ command and $\chi = 1$, then it uses the final round message from the external challenger for the semi-honest protocol and generates $x_0^b = m_b \oplus s_0^{c_0}$, $x_1^b = m_b \oplus s_1^{c_1}$, $x_0^{1-b} = m_{1-b} \oplus s_0^{1-c_0}$, and $x_1^{1-b} = m_{1-b} \oplus s_1^{1-c_1}$. It generates the view of the adversary in either case and runs the distinguisher between Hyb_0 and Hyb_1 on this view. The reduction outputs whatever the distinguisher outputs.

We now analyse the success probability of the reduction in breaking the semi-honest sender security of the OT protocol. We first observe that the view of the adversary in the first $r - 1$ rounds is independent of the choice of χ. This is because irrespective of the challenge string that is chosen by the challenger, the view of the adversary in both the cases are identical. Thus, the probability that χ incorrectly predicts whether the adversary outputs the $\mathsf{corrupt}$ command is $1/2$. If χ correctly predicts whether the adversary issues the $\mathsf{corrupt}$ command or not, then the view of the adversary is identical to Hyb_1 if the challenger chose $(\overline{s}_0^{1-c_0}, \overline{s}_1^{1-c_1})$ and otherwise, it is identical to Hyb_0. Thus, if the distinguisher between Hyb_0 and Hyb_1 correctly predicts the challenge distribution with probability at least $1/2 + \mu(\lambda)$ (where $\mu(\lambda)$ is non-negligible), then the probability that the reduction correctly predicts the challenge sender input in the OT protocol is at least $1/2 \times 1/2 + 1/2 \times (1/2 + \mu(\lambda)) = 1/2 + \mu(\lambda)/2$. This is a contradiction to the semi-honest sender security of the OT protocol. $\square$

A.3.2 Weak Adaptive Semi-Honest Receiver Security

We start with the description of Sim_R.

Description of Sim_R. Sim_R chooses two uniform random bits c_0, c_1. It begins interacting with the adversary by honestly generating the receiver messages in two OT protocol instances using the choice bits c_0 and c_1 respectively. If the adversary issues a $\mathsf{corrupt}$ command in the first $r-2$ rounds, then the simulator reveals the random tape (which includes the random choice bits c_0, c_1) in the OT protocol to $\mathcal{A}$. To generate the pre-final round message, Sim_R generates the pre-final round message of the two OT protocol instances, it chooses a random bit b' and then sends $d_0 = b' \oplus c_0$ and $d_1 = 1 \oplus b' \oplus c_1$ to the adversary. If the adversary issues the $\mathsf{corrupt}$ command after receiving this message, then Sim_R obtains the actual receiver input b. It sets $\beta = b \oplus b'$. It outputs the secret key which comprises of β, c_β, the secret key of the β-th execution and the bit b.

Proof of Indistinguishability. We show that the outputs of Real_R and Ideal_R are computationally indistinguishable via a hybrid argument.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the output of Real_R.
- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we make a syntactic change where we choose β (used in the secret key sk) uniformly at random before the protocol execution begins rather than choosing it when the pre-final message is generated. This hybrid is identical to the previous hybrid.
- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we make the following changes.
  1. We choose a random bit $\beta \in \{0, 1\}$.
  2. If the adversary does not issue the $\mathsf{corrupt}$ command in the first $r-2$ rounds, then we generate the $(r-1)$-th round message by choosing $d_{1-\beta}$ uniformly instead of computing it as $b \oplus c_{1-\beta}$.
  3. If the adversary issues a corrupt command after sending the $(r-1)$-th round message, we output the secret key to be β, c_β and the secret key of the β-th OT execution along with b.

  We argue in Lemma A.3 that $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$ based on the semi-honest receiver security of the OT protocol.
- $\underline{\mathsf{Hyb}_3}$: In this hybrid, instead of choosing $d_{1-\beta}$ uniformly, we set it to be $1 \oplus b \oplus c_{1-\beta}$ (where $c_{1-\beta}$ is the choice bit of the $(1-\beta)$-th OT execution). Via an identical argument to Lemma A.3, we can show that $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$ based on the semi-honest receiver security of the OT protocol.

  We now note that via a renaming of variables, Hyb_3 is identically distributed to Ideal_R.

Lemma A.3. *Assuming the semi-honest receiver security of the OT protocol, we have* $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$.

Proof. Assume for the sake of contradiction that Hyb_2 and Hyb_1 are computationally distinguishable. We give a reduction to the semi-honest receiver security of the underlying OT protocol.

The reduction chooses a random bit β. It then interacts with an external challenger that generates the messages in a single OT execution (specifically, the interaction corresponding to $(1-\beta)$). It chooses random strings (s^0_β, s^1_β). It chooses random strings $(s^0_{1-\beta}, s^1_{1-\beta})$ as the challenge sender inputs and sends them to the external challenger. It chooses a random bit c_β. It chooses two random bits $(c_{1-\beta}, c'_{1-\beta})$ and provides them as the challenge receiver inputs to the external challenger. It obtains the random tape $s_{1-\beta}$ of the sender for the $(1-\beta)$-th OT execution. It chooses a uniform random tape s_β for the β-th OT execution. It sends $((s_0, s_0^0, s_0^1), (s_1, s_1^0, s_1^1))$ as the sender random tape to the adversary. The adversary outputs the sender inputs (m_0, m_1). The reduction then chooses a uniform bit χ (as the guess of whether the adversary will issue the $\mathsf{corrupt}$ command or not). If $\chi = 0$ (indicating that the adversary issues the $\mathsf{corrupt}$ command), then it generates the messages to be sent to the adversary honestly using c_0, c_1 as the receiver inputs in the two OT executions. If $\chi = 1$, then it uses c_β as the receiver input in the β-th OT execution and generates these messages honestly but for the $(1-\beta)$-th OT execution, it forwards the messages received from the external challenger to the adversary. Before sending the pre-final round message, if the adversary issues a $\mathsf{corrupt}$ command but $\chi = 1$, or if adversary does not issue the $\mathsf{corrupt}$ command but $\chi = 0$, then the reduction outputs a random bit to the external challenger and aborts the interaction with the adversary. On the other hand, if adversary issues a $\mathsf{corrupt}$ command and $\chi = 0$, then it provides the adversary with random tape (which includes the choice bits c_0, c_1) used to generate the receiver OT messages in the protocol. If adversary does not issue the $\mathsf{corrupt}$ command and $\chi = 1$, then it computes (d_0, d_1) where $d_\beta = c_\beta \oplus b$ and $d_{1-\beta} = c_{1-\beta} \oplus b$. It sends these two bits along with the pre-final round message of the two OT protocol executions. If the adversary issues a $\mathsf{corrupt}$ command after receiving the pre-final round message, the reduction outputs β, c_β and the secret key of the β-th OT execution along with b. It generates the view of the adversary in either case and runs the distinguisher between Hyb_1 and Hyb_2 on this view. The reduction outputs whatever the distinguisher outputs.

We now analyse the success probability of the reduction in breaking the semi-honest receiver security of the OT protocol. We first observe that the view of the adversary in the first $r-2$ rounds is independent of the choice of χ. This is because $c_{1-\beta}$ and $c'_{1-\beta}$ are uniformly chosen (and hence, identically distributed) and thus, whatever bit is chosen by the challenger, the view of the adversary in both the cases are identical. Thus, the probability that χ incorrectly predicts whether the adversary outputs the $\mathsf{corrupt}$ command or not is $1/2$. If χ correctly predicts whether the adversary issues the $\mathsf{corrupt}$ command, then the adversary's view is identical to Hyb_2 if the challenger chose $c'_{1-\beta}$ and otherwise, it is identical to Hyb_1. Thus, if the distinguisher between Hyb_2 and Hyb_1 correctly predicts the challenge distribution with probability $1/2 + \mu(\lambda)$ (where $\mu(\lambda)$ is non-negligible), then the probability that the reduction correctly predicts the challenge receiver input is $1/2 \times 1/2 + 1/2 \times (1/2 + \mu(\lambda)) = 1/2 + \mu(\lambda)/2$. This is a contradiction to the semi-honest receiver security of the OT protocol. $\square$

B Three-Round Robust Semi-Honest Protocol for 3MULTPlus

In this section, we give a three-round robust semi-honest protocol (for the definition, refer to Property (2c) in Building Blocks in Section 7.1.1 and to Figure 4) for computing the 3MULTPlus functionality. We note that this protocol is same as the one given in [PS21] except that we make use of a two-round oblivious transfer with equivocal receiver security given in [IKSS21]. In subsection B.1, we give the formal definition of the two-round oblivious transfer protocol with equivocal receiver security and in subsection B.2, we give the construction, and in the next subsection we give the proof of security.

B.1 Two-Round Oblivious Transfer Protocol with Equivocal Receiver Security

Syntax. Let $\mathsf{OT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{out}_{\mathsf{OT}})$ be a two-round oblivious transfer protocol. The OT_1 algorithm takes in the security parameter 1^λ and the receiver's choice bit b and outputs the first round message otr along with a secret key sk. The OT_2 algorithm takes in the first round message otr, the sender inputs m_0, m_1 and outputs the sender message ots. The $\mathsf{out}_{\mathsf{OT}}$ algorithm takes in the sender message ots and the secret key sk and outputs the message m_b. We say that the OT protocol is a two-round oblivious transfer with equivocal receiver security [GS18, IKSS21, PS21] if it satisfies the following properties:

- **Correctness:** For every input b of the receiver and m_0, m_1 of the sender:
$$\Pr[\mathsf{out}_{\mathsf{OT}}(\mathsf{ots}, (b, sk)) = m_b] = 1$$
where $(\mathsf{otr}, sk) \leftarrow \mathsf{OT}_1(1^\lambda, b)$ and $\mathsf{ots} \leftarrow \mathsf{OT}_2(\mathsf{otr}, m_0, m_1)$.
- **Equivocal Receiver Security.** There exists a special algorithm $\mathsf{Sim}^{Eq}_{\mathsf{OT}}$ that on input 1^λ outputs $(\mathsf{otr}, sk_0, sk_1)$ such that for any $b \in \{0, 1\}$,
$$\{(\mathsf{otr}, sk_b) : (\mathsf{otr}, sk_0, sk_1) \leftarrow \mathsf{Sim}^{Eq}_{\mathsf{OT}}(1^\lambda)\} \approx_c \{(\mathsf{otr}, sk) : (\mathsf{otr}, sk) \leftarrow \mathsf{OT}_1(1^\lambda, b)\}$$
- **Weak Adaptive Semi-Honest Sender Security:** We require the OT protocol to satisfy weak adaptive semi-honest sender security (described in Definition A.1).

We note that any two-round weak adaptive semi-honest oblivious transfer (see Section A.1) is a two-round oblivious transfer with equivocal receiver security. Hence, we get the following corollary.

Corollary B.1. *Assuming the existence of a two-round semi-honest oblivious transfer protocol. Then, there exists a fully black-box construction of two-round special oblivious transfer protocol.*

B.2 Construction

$\mathsf{3MULTPlus}$ **Functionality.** This is a three-party functionality which takes in (x_1, y_1) from party P_1, (x_2, y_2) from party P_2, (x_3, y_3) from party P_3 where for each $i \in [3]$, x_i and y_i are bits. The functionality outputs $(x_1 \cdot x_2 \cdot x_3 \oplus y_1 \oplus y_2 \oplus y_3)$. We want to design a protocol for the $\mathsf{3MULTPlus}$ functionality that has publicly decodable transcript so that each party that obtains the transcript can learn the output of this functionality.

Description of the Protocol. We give the formal description of the construction in Figure 11. This protocol is exactly the same as the one given in [PS21] except that we use a two-round special oblivious transfer instead of any two-round semi-honest secure OT protocol.

B.3 Proof of Security

The correctness of the protocol was shown in [PS21]. We now give the proof of security starting with the description of simulator.

Description of Simulator. We give the formal description of the simulator Sim below. Here, let H be the set of honest parties and let M be the set of corrupted parties.

1. **Round-1 Message from Sim.** To generate the round-1 message from honest parties, Sim does the following:

Round-1: In the first round,

- P_1 computes $(\mathsf{otr}, sk) := \mathsf{OT}_1(1^\lambda, x_1)$.
- P_2 chooses random bits $x_{2,0}, x_{2,1} \leftarrow \{0,1\}$ subject to $x_2 = x_{2,1} + x_{2,0}$. It computes $(\mathsf{otr}_0, sk_0) := \mathsf{OT}_1(1^\lambda, x_{2,0})$ and $(\mathsf{otr}_1, sk_1) := \mathsf{OT}_1(1^\lambda, x_{2,1})$.
- P_3 computes $(\mathsf{otr}_3, sk_3) := \mathsf{OT}_1(1^\lambda, x_3)$.
- P_1 broadcasts otr, P_2 broadcasts $(\mathsf{otr}_0, \mathsf{otr}_1)$, and P_3 broadcasts otr_3.
- 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,

- P_2 computes $\mathsf{ots} \leftarrow \mathsf{OT}_2(\mathsf{otr}, (x_{2,0}, sk_0), (x_{2,1}, sk_1))$. 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 $\mathsf{ots}_3 \leftarrow \mathsf{OT}_2(\mathsf{otr}_3, x_{2,0,0}, x_{2,0,1})$.
- 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 $\mathsf{ots}_b \leftarrow \mathsf{OT}_2(\mathsf{otr}_b, x_{3,0}, x_{3,1})$. It then computes $\overline{\mathsf{ots}} \leftarrow \mathsf{OT}_2(\mathsf{otr}, \mathsf{ots}_0, \mathsf{ots}_1)$.
- 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.

Round-3: In the last round,

- For each $i \in [3]$, P_i computes $\delta_i = \delta_i^1 + \delta_i^2 + \delta_i^3$.
- P_2 sets $z_2 := x_{2,0,0} + y_2 + \delta_2$.
- P_3 computes $x_{2,0,x_3} := \mathsf{out}_{\mathsf{OT}}(\mathsf{ots}_3, (x_3, sk_3))$ and sets $z_3 = x_{2,0,x_3} + x_{3,0} + y_3 + \delta_3$.
- P_1 computes $(x_{2,x_1}, sk_{x_1}) := \mathsf{out}_{\mathsf{OT}}(\mathsf{ots}, (x_1, sk))$ and $\mathsf{ots}_{x_1} := \mathsf{out}_{\mathsf{OT}}(\overline{\mathsf{ots}}, (x_1, sk))$. It then computes $x_{3,x_{2,x_1}} := \mathsf{out}_{\mathsf{OT}}(\mathsf{ots}_{x_1}, (x_{2,x_1}, sk_{x_1}))$. It then sets $z_1 := x_{3,x_{2,x_1}} + y_1 + \delta_1$.
- 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 11: Description of the three-round inner protocol taken verbatim from [PS21]

(a) If $P_1 \in H$, then Sim computes $(\mathsf{otr}, sk_0', sk_1') \leftarrow \mathsf{Sim}_{\mathsf{OT}}^{Eq}(1^\lambda)$.

(b) If $P_2 \in H$, then for each $b \in \{0,1\}$, Sim computes $(\mathsf{otr}_b, sk_{b,0}', sk_{b,1}') \leftarrow \mathsf{Sim}_{\mathsf{OT}}^{Eq}(1^\lambda)$.

(c) If $P_3 \in H$, then Sim computes $(\mathsf{otr}_3, sk_0'', sk_1'') \leftarrow \mathsf{Sim}_{\mathsf{OT}}^{Eq}(1^\lambda)$.

(d) It sends the above computed messages on behalf of the honest parties to the adversary.

2. **Round-1 Message from $\mathcal{A}$.** The adversary generates the round-1 message on behalf of the corrupted parties and sends their inputs.

3. **Round-2 Message from Sim.** Sim receives the adversarial parties inputs and their random tapes. If the first round message from $\mathcal{A}$ is inconsistent or if $\mathcal{A}$ issues the corrupt command, then Sim receives the inputs of the honest parties, and computes the appropriate secret keys for the first round OT message using these inputs and then completes the rest of the protocol as described in Figure 11. On the other hand, if the first round message from $\mathcal{A}$ is consistent and no corrupt command was issued, then Sim does the following:

   (a) If $P_2 \in H$ and if $P_1 \in M$, it sets $\mathsf{ots} \leftarrow \mathsf{OT}_2(\mathsf{otr}, (x_{2,x_1}, sk_{x_1}), (x_{2,x_1}, sk_{x_1}))$. Similarly, it sets ots_3 to be equal to $\mathsf{OT}_2(\mathsf{otr}_3, x_{2,0,x_3}, x_{2,0,x_3})$ if $P_3 \in M$ where x_{2,x_1} and $x_{2,0,x_3}$ are uniformly chosen random bits.

(b) If $P_3 \in H$ and if $P_1 \in M$, then it computes $\mathsf{ots}_{x_1} \leftarrow \mathsf{OT}_2(\mathsf{otr}_{x_1}, x_{3,x_{2,x_1}}, x_{3,x_{2,x_1}})$ where $x_{3,x_{2,x_1}}$ is uniformly chosen. It then sets $\overline{\mathsf{ots}} \leftarrow \mathsf{OT}_2(\mathsf{otr}, \mathsf{ots}_{x_1}, \mathsf{ots}_{x_1})$.

4. **Round-3 Message from Sim.** On receiving the round-2 message from $\mathcal{A}$, if the messages are inconsistent or a corrupt command was issued, then Sim obtains the inputs of all the honest parties. It chooses the appropriate secret keys and then computes the last round messages exactly as described in the protocol. If the adversarial protocol message is consistent and no corrupt command was issued, then Sim obtains the output z of the 3MULTPlus functionality. It computes $\{z_i\}_{i \in M}$ using the transcript and the random tape of the adversary and chooses $\{z_i\}_{i \in H}$ uniformly such that $\oplus_{i \in H} z_i = z \oplus \oplus_{i \in M} z_i$. It then sends $\{z_i\}_{i \in H}$ to $\mathcal{A}$.

Proof of Indistinguishability. We now show that the simulated interaction is indistinguishable to the real world interaction via a hybrid argument. This proof is mostly taken verbatim from [PS21].

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the view of the adversary and the outputs of the honest parties in the real world execution of the protocol.
- $\underline{\mathsf{Hyb}_1}$: Skip this hybrid if $P_2 \notin H$. In this hybrid, we make the following changes:
  1. We receive the first round messages along with the input and the randomness pair from the corrupted parties.
  2. If the first round messages are consistent with the adversarial inputs and the provided random tapes and if no corrupt command was issued, if $P_1 \in M$, we set $\mathsf{ots} \leftarrow \mathsf{OT}_2(\mathsf{otr}, (x_{2,x_1}, sk_{x_1}), (x_{2,x_1}, sk_{x_1}))$ instead of $\mathsf{OT}_2(\mathsf{otr}, (x_{2,0}, sk_0), (x_{2,1}, sk_1))$. Similarly, if $P_3 \in M$, we set ots_3 to be equal to $\mathsf{OT}_2(\mathsf{otr}_3, x_{2,0,x_3}, x_{2,0,x_3})$ instead of $\mathsf{OT}_2(\mathsf{otr}_3, x_{2,0,0}, x_{2,0,1})$.

  This hybrid is computationally indistinguishable to the previous hybrid from the weak adaptive sender security of the special OT protocol.
- $\underline{\mathsf{Hyb}_2}$: Skip this hybrid if $P_3 \notin H$. In this hybrid, if the first round messages from $\mathcal{A}$ are consistent and if no corrupt command was issued, then for each $b \in \{0, 1\}$, we set $\mathsf{ots}_b \leftarrow \mathsf{OT}_2(\mathsf{otr}_b, x_{3,x_{2,b}}, x_{3,x_{2,b}})$ instead of $\mathsf{OT}_2(\mathsf{otr}_b, x_{3,0}, x_{3,1})$. If $P_1 \in M$, we then set $\overline{\mathsf{ots}} \leftarrow \mathsf{OT}_2(\mathsf{otr}, \mathsf{ots}_{x_1}, \mathsf{ots}_{x_1})$. This hybrid is again computationally indistinguishable to the previous hybrid from the weak-adaptive sender security of the special OT protocol.
- $\underline{\mathsf{Hyb}_3}$: Skip this hybrid change if $P_1 \notin H$. In this hybrid, we compute $(\mathsf{otr}, sk'_0, sk'_1) \leftarrow \mathsf{Sim}^{Eq}_{\mathsf{OT}}(1^\lambda)$ and set $sk = sk'_{x_1}$. We continue with the rest of the execution as before. This hybrid is computationally indistinguishable to Hyb_0 from the equivocal receiver security of the OT protocol.
- $\underline{\mathsf{Hyb}_4}$: Skip this hybrid if $P_3 \notin H$. In this hybrid, we compute $(\mathsf{otr}_3, sk'_0, sk'_1) \leftarrow \mathsf{Sim}^{Eq}_{\mathsf{OT}}(1^\lambda)$ and set $sk_3 = sk'_{x_3}$. This hybrid is computationally indistinguishable to the previous hybrid from the equivocal receiver security of the OT protocol.
- $\underline{\mathsf{Hyb}_5}$: Skip this hybrid if $P_2 \notin H$. In this hybrid, for each $b \in \{0, 1\}$, we compute $(\mathsf{otr}_b, sk'_{b,0}, sk'_{b,1}) \leftarrow \mathsf{Sim}^{Eq}_{\mathsf{OT}}(1^\lambda)$. We then set $sk_0 = sk'_{0,x_{2,0}}$ and $sk_1 = sk'_{1,x_{2,1}}$. This hybrid is again computationally indistinguishable to the previous hybrid from the equivocal receiver security of the OT protocol.
- $\underline{\mathsf{Hyb}_6}$: Let i^* be the smallest integer such that $P_{i^*} \in H \cap \{P_1, P_2, P_3\}$. In this hybrid, if the messages received from adversary in the first two rounds are consistent and if no corrupt

command was issued, then we set $z_{i^*} = z - \sum_{j \in [3] \setminus \{i^*\}} z_j$ instead of computing it as in the previous hybrid. Here, z is the output of the ideal functionality. This change is again syntactic and hence, this hybrid is identical to the previous one.

- $\underline{\mathsf{Hyb}_7}$: If the messages received from adversary in the first two rounds are consistent and if no corrupt command was issued, then for every $i \in H \cap \{P_1, P_2, P_3\}$ and $i \neq i^*$, we choose z_i uniformly at random. This hybrid is identically distributed to the previous one since $\delta_1, \delta_2, \delta_3$ form an additive secret sharing of 0. Note that Hyb_7 is identical to the simulated distribution.