Round-Optimal Black-Box Protocol Compilers

Yuval Ishai[1], Dakshita Khurana[2], Amit Sahai[3], and Akshayaram Srinivasan[4]

[1] Technion
[2] UIUC
[3] UCLA
[4] Tata Institute of Fundamental Research

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 OT correlations model. We use our compilers to obtain the following results:

- A two-round, two-party protocol secure against malicious adversaries 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 considering special functionalities such as a two-round oblivious transfer. This result also implies the first constructions of two-round malicious (batch) OT/OLE in the random oracle model based on the black-box use of two-round semi-honest (batch) OT/OLE.
- A three-round multiparty secure computation protocol in the random oracle model secure against malicious adversaries that is based on the black-box use of two-round semi-honest OT. This protocol matches a known round complexity lower bound due to Applebaum et al. (ITCS'20) and is based on a minimal cryptographic hardness assumption.
- A two-round, multiparty secure computation protocol in the 1-out-of-2 OT correlations model that is secure against malicious adversaries and makes black-box use of cryptography. This gives new round-optimal protocols for computing arithmetic branching programs that are statistically secure and makes 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. [33] proposed concretely efficient 2-round oblivious transfer (OT) protocols under several standard assumptions. Other concretely efficient 2-round OT protocols were proposed in [26, 27]. Chase et al. [10] and Branco et al. [7] 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 [6, 5]. 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. [5] 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 without increasing round complexity, without strengthening the underlying assumptions, and *without significantly hurting concrete efficiency*. A clean theoretical model for capturing the latter is a *black-box construction*. Such a construction builds a malicious-secure protocol by using an underlying semi-honest protocol *as an oracle*. The latter 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 [11] (replacing an earlier non-black-box construction from [4]), a black-box construction of malicious-secure OT from semi-honest OT [18] or trapdoor permutations [29] (replacing a non-black-box construction of [17]), and a black-box construction for OT extension [20] (replacing the earlier non-black-box protocol [3]).

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 [19], building upon the IPS compiler of [24], 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;[5] and *(ii) adaptive security with erasures*, a limitation inherited from [24]. This latter property is typically easy to achieve by increasing the round complexity. However, it poses a major challenge in the 2-round setting. While natural two-round protocols *in 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*?

1.1 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 [19] by standard OT correlations. We now give a more detailed account of our results.

Round-Preserving Compilers in the OT Correlations Model. In the case of two-party protocols in the OT correlations model, we remove the need for adaptive security with erasures and obtain the following result.

Informal Theorem 1 *There exists a black-box compiler from any two-round semi-malicious two-party protocol to a two-round malicious two-party computation protocol given a setup that consists of random* 1*-out-of-*2 *OT correlations (alternatively, Rabin-OT correlations).*

See Theorem 3 for a formal statement. As in the case of the IPS compiler [24], 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 [34]). But when f is a "simple" functionality such as batch-OT[6] or batch-OLE, we can in fact use f' that consists of only a *constant* number of instances of f.

[5] 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.

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

We note that the required setup 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 [6] can make the setup reusable without additional interaction.

To obtain this result, we build a *new* version of the black-box protocol compiler of [24], where we replace the outer protocol with one that can be simpler and more efficient than the state-of-the-art [23] protocol previously used in this setting. Besides eliminating the need for adaptive security from the semi-honest MPC protocol, the improved outer protocol may be of independent interest.

However, our primary contribution (that also uses the techniques developed above) is the construction of round-optimal compilers in the Random Oracle model, as we discuss next.

Round-Preserving 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 [24]). 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 perform the watchlist function. This gives rise to new round-optimal malicious secure protocols in the random oracle model from black-box use of semi-honest secure protocol.[7]

The Two-Party Setting. We obtain the following results in the two-party setting. Here, non-interactive secure computation (NISC) denotes a two-round 2-party secure computation protocol for general functionalities where only one party obtains an output. A two-sided non-interactive secure computation (NISC) denotes a two-round 2-party secure computation protocol for general functionalities where both parties obtain an output.

Informal Theorem 2 (BB Malicious NISC) *There exists a construction of NISC with malicious security in the random oracle model that makes black-box use of NISC with semi-honest security.*

Informal Theorem 3 (BB Malicious 2-sided NISC) *In the random oracle model, there exists a construction of two-sided NISC with malicious security that makes black-box use of two-sided NISC with semi-honest security.*

As before, the functionality computed by the semi-honest protocol depends on the target functionality computed by the malicious protocol. For the case of simple functionalities such as (batch)-OT and (batch)-OLE, these two functions are identical. The formal statement of the transformation in the random oracle

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

model can be found in Theorem 2 and its extension to the two-sided setting appears in Section 5.3.

We note that [28] also used the Fiat-Shamir transform to collapse the number of rounds of a NISC 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 [34]. This implies the following corollaries of Informal Theorem 2:

Informal Corollary 1 *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 2 *There exists a construction of two-round OLE/batch OT/batch OLE respectively with malicious security in the random oracle model that makes black-box use of two-round OLE/batch OT/batch OLE respectively with semi-honest security.*

Prior to our work, the only known construction of two-round malicious OLE relied on specialized assumptions such as N^{th} residuosity [10] or LWE [7]. The black-box constructions of OT required assumptions stronger than semi-honest security in the random oracle model [26, 27] or in the plain model [13] (such as strongly uniform key agreement).

Protocol Compilers in the Multi-Party 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 the minimal cryptographic hardness assumption which is a two-round semi-honest OT protocol.

Informal Theorem 4 *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 4. Applebaum et al.[1] 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 [32] gave a construction of a three-round malicious secure protocol from any two-round oblivious transfer that satisfied a certain form of adaptive security on the receiver side. In this work, we construct a malicious secure protocol by relying only a two-round semi-honest OT (in the random oracle model).

As an additional contribution, we show how to remove the complex multi-party watchlist correlations setup from the work of [19] and replace it with a simple 1-out-of-2 random OT correlations setup. As a corollary, this gives the first constructions of statistical secure protocols against malicious adversaries

for computing arithmetic branching programs making black-box use of the underlying field in the OLE correlations model. The formal statement appears in Theorem 5.

2 Technical Overview

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

2.1 IPS Compiler

The starting point of our work is the black-box compiler given by Ishai, Prabhakaran, and Sahai [24] (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 [23] (see also [30]) making black-box use of a PRG. This protocol is referred to as the *outer protocol*.
- A semi-honest secure[8] 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. However, a malicious adversary can cheat in all the inner protocol instances, thereby breaking the security of each one of them. Note that the outer protocol is only guaranteed to be secure as long as a constant fraction of the servers are corrupted. To ensure this property, the IPS compiler uses a special "cut-and-choose" mechanism referred to as *watchlists*.

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

The simplest version of 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 [24], 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 as *equivocal receiver security* [15]. 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 is 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 adversary, 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 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.

2.2 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. [16] 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 a 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 extremely efficient outer protocols that also satisfy output error correction. In particular, we show that the semi-honest secure protocol from [21] is secure against pairwise verifiable adversaries if we replace the plain Shamir secret sharing with a bi-variate Shamir secret sharing. 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 rounds. To see why this is the case, we modify the watchlist mechanism checks 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.

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 [8].
2. We then use the transformation described in [24, 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.

2.3 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 [12] 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[9]. 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 if the openings are correct, if the random oracle output is correctly computed, if the input–randomness pair in the opened executions are

[9] Such a commitment can be constructed unconditionally in the random oracle model [31].

consistent with the transcript seen so far, and if the pairwise consistency checks pass.

In the security proof, we rely on the correlation-intractability of the random oracle [9] 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.

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.

2.4 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 [19]), 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.

2.5 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 [1] gave some barriers). Moreover, in existing three-round protocols [32], 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 [32] 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 [32] 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 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.

3 Preliminaries

Let λ denote the cryptographic security parameter. We assume that all cryptographic algorithms implicitly take 1^λ as input. A function $\mu(\cdot) : \mathbb{N} \to \mathbb{R}^+$ is said to be negligible if for any polynomial $\mathsf{poly}(\cdot)$, there exists λ_0 such that for

all $\lambda > \lambda_0$, we have $\mu(\lambda) < \frac{1}{\mathsf{poly}(\lambda)}$. We will use $\mathsf{negl}(\cdot)$ to denote an unspecified negligible function and $\mathsf{poly}(\cdot)$ to denote an unspecified polynomial function.

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

3.1 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 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 1. 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.

3.2 Semi-Malicious Two-Round Two-Party Computation

Semi-Malicious security [2] 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 2 (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 semi-malicious adversary $\mathcal{A}$ corrupting either the sender or the receiver and for any receiver's input x, sender's input y and for any random tape r, we have:*

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

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

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

3.3 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 [31].

3.4 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 3 (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 (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 $\mathsf{Extrapolate}$ that on input t', $\{x_i\}_{i \notin T}$ outputs $\{x'_i\}_{i \in T}$ such that $(\{x_i\}_{i \notin T}, \{x'_i\}_{i \in T})$ belongs to $\mathsf{supp}(\mathsf{Share}_{(t',m)}(x))$.*

We note that the above definition of pairwise verifiable secret sharing is the same as the one given in [23] except that 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 $m \geq kt + 2\ell + 1$.

4 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 4.1.

4.1 Definition

Syntax. Let f be an arbitrary n-party functionality. Consider the standard client-server MPC setting [11] 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 (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 6 (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) 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) 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.*

We state the main theorem about constructing pairwise verifiable MPC protocol and defer the proof to the full version.

Theorem 1. *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). 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 6) 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.*

5 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 5.1, we give our compiler in the random oracle model. In Section 5.2, we give our compiler in the OT correlations model. Finally, in Section 5.3, we show how to extend these compilers to give a round-optimal, malicious-secure, two-party protocol in the two-sided setting.

5.1 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 2. *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 6).*
- *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 1) where* Π_i *computes the function* $\mathsf{Eval}(i, \cdot)$*.*

Then, 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*.*

Instantiating the pairwise verifiable MPC protocol from Theorem 1, we get the following corollary.

Corollary 1. *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 5.1, we describe the construction of the above malicious-secure protocol and in section 5.1, we give the proof of security.

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 6). 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 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 3.3).
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.

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.

– **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.

Fig. 1: Description of r-round Malicious 2PC

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 P_i to the corrupted servers.

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.

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 $\{(x_j^i, s_j^i), \mathsf{Open}(\mathsf{com}_j^i)\}$ from $\mathbb{T}$.
   (b) It checks if $\mathsf{Open}(\mathsf{com}_j^i)$ is valid.
   (c) It then checks if $(x_j^i, H_1(i, j, x_j^i, s_j^i))$ is a valid (input,randomness) pair for the protocol Π_j consistent with the transcript $\mathbb{T}$.
   (d) For each $j' \in K_i$, it checks if $P(i, j, j', x_j^i, x_{j'}^i) = 1$.
3. If any of the checks fail, it outputs 0. Else, if all the checks pass, it outputs 1.

Fig. 2: Description of chkConsistency

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.
8. For each $j \in [m]$, if $(x_j^{1-i}, H_1(1-i, j, x_j^{1-i}, s_j^{1-i}))$ 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_j^i. 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 $P(1-i, j, k, x_j^{1-i}, x_k^{1-i}) = 0$. 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 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 chkConsistency then Sim sends $\perp$ to the trusted functionality.
12. 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.
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 $(x_j^0, H_1(0, j, x_j^0, s_j^0))$ 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.

- $\underline{\mathsf{Real}}$: This corresponds to the output of the real execution of the protocol.
- $\underline{\mathsf{Hyb}_0}$: This hybrid corresponds to the distribution where the 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.
- $\underline{\mathsf{Hyb}_1}$: 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 1, we show that Hyb_0 and Hyb_1 are statistically indistinguishable from the error correction properties of Φ (see Definition 4.1).
- $\underline{\mathsf{Hyb}_2}$: 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.
- $\underline{\mathsf{Hyb}_3}$: 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 2, we show that $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$ from the semi-honest sender security of the inner protocol.

– 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 3, 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 1. *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.

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

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. [22, 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 $\mathsf{Hyb}_2 \approx_s \mathsf{Hyb}_3$.

Lemma 2. *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 the this set when queried on the messages generated by P_i.

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 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 3. *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_2 and Hyb_3 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.

5.2 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 3. *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 6).*
- *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 2) where Π_i computes the function $\mathsf{Eval}(i, \cdot)$.*

Then, 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.

Instantiating the pairwise verifiable MPC protocol from Theorem 1, we get the following corollary.

Corollary 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 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.*

We defer the proof of Theorem 3 to the full version.

5.3 Extension to the Two-Sided Setting

In this subsection, we explain how to extend the protocol described in Section 5.1 to the bidirectional communication model. Specifically, we want to construct an two-round 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.

We defer the proof of security of this construction to the full version.

6 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.

6.1 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 [1] 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, [32] gave a malicious-secure construction in the CRS model assuming a two-round malicious secure oblivious transfer protocol that additionally satisfies equivocal receiver security [15].

We give the formal statement of our theorem below.

Theorem 4. *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 6).*
– *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 1, we get the following corollary.

Corollary 3. *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*.*

We give the proof of Theorem 4 in the full version.

6.2 Protocol Compiler in the OT Correlations Model

In this subsection, we improve the result from [19] and give a construction of a two-round black-box protocol for computing multiparty functionalities that are secure 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 [19] and recalled in Definition 7).

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[10] 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 [23, 30] by making black-box use of a PRG. As noted in [19], 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.

[10] 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 [23] for the formal definition.

Definition 7 ([19]). *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:*

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

  where the real and ideal experiments are described in Figure 3 and for each $i \in H$, r_i is uniformly chosen.

$\mathsf{Real}(\mathcal{A}, \{x_i, r_i\}_{i \in H})$	$\mathsf{Ideal}(\mathcal{A}, \mathsf{Sim}_\Pi, \{x_i\}_{i \in H})$
(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 $\{\mathsf{out}_\Pi(i, \pi(2))\}_{i \in H}$.

Fig. 3: Security Game for the Two-Round Inner Protocol

[19] showed that the protocol from [14] in the OT correlations model and [25] 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 5. *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.*

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.

We give the proof of this theorem in the full version.

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 the SERB startup grant.

References

1. Applebaum, B., Brakerski, Z., Garg, S., Ishai, Y., Srinivasan, A.: Separating two-round secure computation from oblivious transfer. In: ITCS 2020. LIPIcs, vol. 151, pp. 71:1–71:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020), https://doi.org/10.4230/LIPIcs.ITCS.2020.71
2. Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., Wichs, D.: Multiparty computation with low communication, computation and interaction via threshold FHE. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 483–501. Springer, Heidelberg, Germany, Cambridge, UK (Apr 15–19, 2012)

3. Beaver, D.: Correlated pseudorandomness and the complexity of private computations. In: Miller, G.L. (ed.) STOC 96. pp. 479–488. ACM (1996), https://doi.org/10.1145/237814.237996
4. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: 22nd ACM STOC. pp. 503–513. ACM Press, Baltimore, MD, USA (May 14–16, 1990)
5. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Rindal, P., Scholl, P.: Efficient two-round OT extension and silent non-interactive secure computation. In: CCS 2019. pp. 291–308. ACM (2019), https://doi.org/10.1145/3319535.3354255
6. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudorandom correlation generators: Silent OT extension and more. pp. 489–518. LNCS, Springer, Heidelberg, Germany, Santa Barbara, CA, USA (2019)
7. Branco, P., Döttling, N., Mateus, P.: Two-round oblivious linear evaluation from learning with errors. IACR Cryptol. ePrint Arch. p. 635 (2020), https://eprint.iacr.org/2020/635
8. Brassard, G., Crépeau, C., Robert, J.M.: Information theoretic reductions among disclosure problems. In: 27th FOCS. pp. 168–173. IEEE Computer Society Press, Toronto, Ontario, Canada (Oct 27–29, 1986)
9. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM 51(4), 557–594 (2004), https://doi.org/10.1145/1008731.1008734
10. Chase, M., Dodis, Y., Ishai, Y., Kraschewski, D., Liu, T., Ostrovsky, R., Vaikuntanathan, V.: Reusable non-interactive secure computation. pp. 462–488. LNCS, Springer, Heidelberg, Germany, Santa Barbara, CA, USA (2019)
11. Damgård, I., Ishai, Y.: Constant-round multiparty computation using a black-box pseudorandom generator. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 378–394. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2005)
12. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 1987)
13. Friolo, D., Masny, D., Venturi, D.: A black-box construction of fully-simulatable, round-optimal oblivious transfer from strongly uniform key agreement. In: Hofheinz, D., Rosen, A. (eds.) TCC. Lecture Notes in Computer Science, vol. 11891, pp. 111–130. Springer (2019), https://doi.org/10.1007/978-3-030-36030-6_5
14. Garg, S., Ishai, Y., Srinivasan, A.: Two-round MPC: Information-theoretic and black-box. In: TCC 2018, Part I. pp. 123–151. LNCS, Springer, Heidelberg, Germany (Mar 2018)
15. Garg, S., Srinivasan, A.: Two-round multiparty secure computation from minimal assumptions. pp. 468–499. LNCS, Springer, Heidelberg, Germany (2018)
16. Gennaro, R., Ishai, Y., Kushilevitz, E., Rabin, T.: On 2-round secure multiparty computation. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 178–193. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2002)
17. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A. (ed.) 19th ACM STOC. pp. 218–229. ACM Press, New York City, NY, USA (May 25–27, 1987)
18. Haitner, I., Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: Black-box constructions of protocols for secure computation. SIAM J. Comput. 40(2), 225–266 (2011), https://doi.org/10.1137/100790537

19. Ishai, Y., Khurana, D., Sahai, A., Srinivasan, A.: On the round complexity of black-box secure MPC. In: CRYPTO 2021. Lecture Notes in Computer Science, vol. 12826, pp. 214–243. Springer (2021), https://doi.org/10.1007/978-3-030-84245-1_8
20. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2003)
21. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In: 41st FOCS. pp. 294–304. IEEE Computer Society Press, Redondo Beach, CA, USA (Nov 12–14, 2000)
22. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Johnson, D.S., Feige, U. (eds.) 39th ACM STOC. pp. 21–30. ACM Press, San Diego, CA, USA (Jun 11–13, 2007)
23. Ishai, Y., Kushilevitz, E., Paskin, A.: Secure multiparty computation with minimal interaction. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 577–594. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 2010)
24. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer - efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2008)
25. Lin, H., Liu, T., Wee, H.: Information-theoretic 2-round MPC without round collapsing: Adaptive security, and more. In: TCC 2020. Lecture Notes in Computer Science, vol. 12551, pp. 502–531. Springer (2020), https://doi.org/10.1007/978-3-030-64378-2_18
26. Masny, D., Rindal, P.: Endemic oblivious transfer. In: CCS 2019. pp. 309–326. ACM (2019), https://doi.org/10.1145/3319535.3354210
27. McQuoid, I., Rosulek, M., Roy, L.: Minimal symmetric PAKE and 1-out-of-n OT from programmable-once public functions. In: CCS 2020. pp. 425–442. ACM (2020), https://doi.org/10.1145/3372297.3417870
28. Mohassel, P., Rosulek, M.: Non-interactive secure 2pc in the offline/online and batch settings. In: Eurocrypt 2017. Lecture Notes in Computer Science, vol. 10212, pp. 425–455 (2017), https://doi.org/10.1007/978-3-319-56617-7_15
29. Ostrovsky, R., Richelson, S., Scafuro, A.: Round-optimal black-box two-party computation. In: CRYPTO 2015, Part II. pp. 339–358 (2015)
30. Paskin-Cherniavsky, A.: Secure Computation with Minimal Interaction. Ph.D. thesis, Technion (2012), available at http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/2012/PHD/PHD-2012-16.pdf
31. Pass, R.: On deniability in the common reference string and random oracle model. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 316–337. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2003)
32. Patra, A., Srinivasan, A.: Three-round secure multiparty computation from black-box two-round oblivious transfer. In: CRYPTO 2021. Lecture Notes in Computer Science, vol. 12826, pp. 185–213. Springer (2021), https://doi.org/10.1007/978-3-030-84245-1_7
33. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2008)
34. Yao, A.C.: How to generate and exchange secrets (extended abstract). In: 27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986. pp. 162–167. IEEE Computer Society (1986), https://doi.org/10.1109/SFCS.1986.25