On the Round Complexity of Black-Box Secure MPC

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

Abstract

We consider the question of minimizing the *round complexity* of secure multiparty computation (MPC) protocols that make a *black-box* use of simple cryptographic primitives in the setting of security against any number of malicious parties. In the plain model, previous black-box protocols required a high constant number of rounds (>15). This is far from the known lower bound of 4 rounds for protocols with black-box simulators.

When allowing a random oblivious transfer (OT) correlation setup, 2-round protocols making a black-box use of a pseudorandom generator were previously known. However, such protocols were obtained via a round-collapsing "protocol garbling" technique that has poor concrete efficiency and makes a non-black-box use of an underlying malicious-secure protocol.

We improve this state of affairs by presenting the following types of black-box protocols.

- **4-round "pairwise MPC" in the plain model.** This round-optimal protocol enables each ordered pair of parties to compute a function of both inputs whose output is delivered to the second party. The protocol makes black-box use of any public-key encryption (PKE) with pseudorandom public keys. As a special case, we get a black-box round-optimal realization of secure (copies of) OT between every ordered pair of parties.
- **2-round MPC from OT correlations.** This round-optimal protocol makes a black-box use of any *general* 2-round MPC protocol satisfying an augmented notion of *semi-honest* security. In the two-party case, this yields new kinds of 2-round black-box protocols.
- **5-round MPC in the plain model.** This protocol makes a black-box use of PKE with pseudorandom public keys, and 2-round oblivious transfer with "semi-malicious" security.

A key technical tool for the first result is a novel combination of split-state non-malleable codes (Dziembowski, Pietrzak and Wichs, JACM '18) with standalone secure *two-party* protocols. The second result is based on a new round-optimized variant of the "IPS compiler" (Ishai, Prabhakaran and Sahai, Crypto '08). The third result is obtained via a specialized combination of these two techniques.

*Technion. Email: yuvali@cs.technion.ac.il

†UIUC. Email: dakshita@illinois.edu

‡UCLA. Email: sahai@cs.ucla.edu

§Tata Institute of Fundamental Research. Email: akshayaram.srinivasan@tifr.res.in. Work done in part while at UC Berkeley.

Contents

1 Introduction

Minimizing the *round complexity* of cryptographic protocols has been a central theme of research in the past few decades. Much of this research focused on the question of minimizing the round complexity of protocols for *secure multiparty computation* (MPC), both in the general case as well as for special tasks of interest such as zero-knowledge proofs, oblivious transfer (OT), or coin-tossing. This question is motivated not only by its direct relevance to the latency of protocols running over real-life networks, but also as an intriguing theoretical challenge that often inspires new ideas and serves as a test bed for new techniques.

The round complexity of MPC. We consider the standard setting of MPC with *an arbitrary number* of *malicious* parties, namely parties that are corrupted by a central adversary who may arbitrarily change their behavior. What do we know about the round complexity of MPC in this setting? Allowing a *common random string* (CRS) setup, it was recently shown [BL18, GS18] that *2-round* MPC protocols are possible under the (minimal) assumption that 2-round OT exists in the CRS model. This round complexity is clearly optimal, even in the easier setting of *semi-honest* adversaries who send messages as instructed by the protocol. In the *plain model*, without any setup, a long line of works [GMPP16, BHP17, ACJ17, KS17, BGJ$^+$17, BGJ$^+$18, HHPV18, CCG$^+$20] has culminated in *4-round* protocols that rely on the minimal assumption that a 4-round OT protocol exists [CCG$^+$20]. This round complexity is known to be optimal for protocols that admit a *black-box simulator* [GK96b, KO04, GMPP16]. All of the above 4-round protocols are of this kind.

Black-box constructions. Another central research theme in cryptography is obtaining *black-box constructions* of higher-level primitives from simpler lower-level primitives. A black-box construction of X from Y, also known as a (fully) black-box *reduction* from X to Y [RTV04], specifies an implementation of X that only has oracle access to the input-output relation of Y, without being given any explicit representation of Y, e.g., in the form of a Boolean circuit or a Turing Machine. Moreover, it is required that the security reduction be black-box in the sense that any adversary A_X "attacking" X can be used as a black-box to obtain an adversary A_Y who obtains a similar advantage in attacking Y. Originating from the pioneering work of Impagliazzo and Rudich [IR90], a long line of works study the landscape of black-box reductions between natural cryptographic primitives. More relevant to our work is the effort to replace known instances of *non-black-box* constructions, where X requires access to the *code* of Y, by black-box constructions.

In the MPC context, early examples of results along this line include a black-box construction of constant-round *honest-majority* MPC protocols from one-way functions [DI05] (replacing an earlier non-black-box construction from [BMR90]) and a black-box construction of malicious-secure OT from semi-honest OT [HIK$^+$11] (replacing a non-black-box construction of [GMW87]). Beyond the theoretical interest in understanding the tightness of the relation between primitives, the goal of replacing non-black-box constructions by black-box counterparts is strongly motivated by asymptotic and concrete *efficiency*. A well-known example in the context of MPC is the non-black-box *OT extension* construction of Beaver [Bea96], which was replaced by a much more efficient black-box construction from [IKNP03] that is commonly used as a basis for fast MPC implementations. We use the term *black-box MPC* to refer generically to an MPC protocol obtained via a black-box construction from simple low-level primitives (such as OT) that can be easily and efficiently constructed from standard cryptographic assumptions.

Round complexity of *black-box* MPC. Interestingly, all of the round-optimal MPC protocols in the standard setting we consider, including those mentioned above, make *non-black-box* use of the underlying primitives. In the case of 2-round MPC protocols in the CRS model, this is known to be inherent (even for the easier goal of semi-honest security), at least for black-box constructions from 2-round OT or any other *2-party* protocol [ABG+20]. However, no such impossibility result is known for 4-round MPC protocols in the *plain model*.

In the two-party case, a 4-round black-box protocol is known for *one-sided* functionalities that deliver output to only one of the two parties [ORS15, FMV19]. The most general protocol of this kind makes a black-box use of any public-key encryption (PKE) with pseudorandom public keys, which can be easily constructed from most standard cryptographic assumptions [FMV19]. This implies a similar 5-round protocol for two-sided functionalities.

In contrast, for a general number of parties, all known constant-round protocols are either complex and inefficient, or resort to idealized models such as the Random Oracle (RO) model to achieve better efficiency but only heuristic security. Despite the significant body of work on the round complexity of black-box MPC and related primitives in the plain model, the best exact round complexity that follows from existing works [IPS08, Wee10, Goy11] is greater than 15 (see Section 1.2). Recent attempts to minimize round complexity [GMPP16, BHP17, BGJ+17, KS17, ACJ17, BGJ+18, HHPV18, CCG+20] have led to complex protocols that make heavy non-black-box use of cryptography. This gap gives rise to the first motivating question for our work.

What is the minimal round complexity of black-box MPC in the plain model?
Must we necessarily resort to idealized models to achieve simplicity and/or efficiency?

Round complexity of black-box protocol transformations. It turns out that if "plain model" is relaxed to allow a simple setup in the form of random *OT correlations* between each pair of parties, the first part of the above question has been settled. Concretely, given an OT correlation setup, which can be generated with good concrete efficiency [IKNP03, BCG+19], there is a 2-round MPC protocol making a black-box use of a pseudorandom generator [GIS18]. However, this 2-round protocol is quite complex and inefficient, as it is obtained by applying a heavily non-black-box "protocol garbling" transformation [GS18, BL18] to an underlying multi-round (information-theoretic) MPC protocol. This not only hurts asymptotic and concrete efficiency, but also rules out applying this transformation while respecting a black-box use of an underlying primitive. The latter includes a black-box use of an algebraic structure (e.g., a big finite field), a cryptographic primitive (e.g., homomorphic encryption or even a random oracle), or an ideal functionality oracle (e.g., OT or its arithmetic variant OLE). This is similar to the classical non-black-box protocol transformation from semi-honest MPC to malicious MPC, due to Goldreich, Micali, and Wigderson [GMW87], which is limited in the same way.

In contrast, "black-box protocol transformations" from weak MPC protocols to stronger ones, commonly known as "MPC-in-the-head" transformations [IKOS07, IPS08], have avoided these limitations. In a nutshell, such transformations obtain a strong MPC protocol for f (say, with malicious security) by making a black-box use of a weak MPC protocol (say, with semi-honest security) for a *related* functionality f'. The relation between f and f' needs to be restricted in some way. Typically, f' is a next-message function of (an information-theoretic) weak MPC protocol for f. This black-box protocol transformation paradigm, systematically studied in [IKP+16], has not only given rise to new theoretical feasibility and efficiency results, but it has also led to practical zero-knowledge proof systems [GMO16, AHIV17], digital signatures [CDG+17, KKW18], and MPC protocols [HIMV19]. The question we ask is whether one can obtain a similar black-box protocol transformation in the context of 2-round MPC with OT correlation setup:

Are there useful kinds of "black-box protocol transformations" from 2-round semi-honest MPC to 2-round malicious MPC with OT correlation setup?

This question is particularly motivated in the two-party case, where there are many different techniques for efficient 2-round semi-honest protocols that make black-box use of algebraic or cryptographic primitives.

1.1 Our Contributions

We make progress on these questions by obtaining the following types of round-efficient black-box protocols.

1.1.1 Black-box 4-round "Pairwise MPC" in the Plain Model.

Our first result addresses the first question by settling the round complexity of black-box MPC for a restricted but useful class of functionalities. Concretely, we get a 4-round black-box protocol for any *pairwise* MPC functionality that enable each ordered pair of parties to simultaneously compute a one-sided function of their inputs, whose output is delivered to the second party. The protocol makes a black-box use of any public-key encryption (PKE) with pseudorandom public keys, similar to the 4-round 2-party OT protocol of [FMV19].

The central challenge in the pairwise MPC setting is to develop two-party protocols that remain secure *when executed in parallel*. We develop new black-box protocols for this setting, starting with the case of OT protocols, and generalizing via the result of [IKO$^+$11] to any two-party functionality. To this end, a technical contribution of our work is a novel combination of split-state non-malleable codes [DPW18, CGL16] with standalone secure *two-party* protocols to obtain black-box, *non-malleable* two-party protocols.

The resulting pairwise MPC can be used to generate OT correlations in a preprocessing phase, as required by the 2-round black-box protocol of [GIS18]. This results in a 6-round MPC protocol making black-box use of PKE with pseudorandom public keys. While this already constitutes a major improvement over the state of the art, it is still two rounds away from the 4-round lower bound. Perhaps more importantly, as discussed above, the [GIS18] approach employs a round-collapsing "protocol garbling" that limits its efficiency and applicability to protocols that make black-box use of algebraic or cryptographic primitives. Motivated by both limitations, we would like to replace the protocol garbling technique by a black-box protocol transformation that takes advantage of OT correlations.

1.1.2 An "IPS-style Compiler" for 2-Round MPC.

Our second main contribution is a new black-box protocol transformation obtained via a round-optimized variant of the "IPS compiler" [IPS08]. This transformation uses a 2-round honest-majority MPC protocol from [IKP10, Pas12] to transform in a black-box way any 2-round MPC protocol with an augmented variant of *semi-honest* security to obtain a 2-round MPC protocol with malicious security. The transformation relies on a special form of OT correlations referred to as *watchlist correlations*. Specifically, for each pair of parties i, j, the ideal watchlist correlation is similar to random k-out-of-m OT in that it gives to party i a random m-tuple of strings and to party j a random subset K_j of k of the m strings. However, the different OT instances are correlated in that the same subset K_j is received by party j in all instances in which it acts as a receiver. In fact, the latter consistency condition should only hold for honest parties j. This gives rise to a simple protocol for generating the watchlist correlation using parallel invocations of OT, which in turn can be implemented using our pairwise OT protocol. Combined with our first main result, this yields the same kind of 6-round black-box protocol obtained via [GIS18], but with the advantage of making a black-box use of an augmented semi-honest protocol (as opposed to a non-black-box use of a malicious protocol incurred by the protocol garbling technique).

The augmented semi-honest security requirement combines the so-called *semi-malicious* security [AJL$^+$12a], which is satisfied by most natural 2-round semi-honest protocols, with a form of *adaptive security with erasures*. The latter is satisfied by all natural information-theoretic protocols (with standard forms of setup), as well as by computationally secure protocols with pre-processing. Concretely, we show the protocol from [GIS18] in the OT correlations model and the protocol from [LLW20] in the OLE correlations model satisfy augmented semi-honest security and thus, can be used in our compiler.

We obtain a couple of interesting corollaries by instantiating our compiler with appropriate inner protocols. As a first corollary, if we instantiate our compiler with the protocol from [GIS18] that has statistical augmented semi-honest security for computing functions in $\mathcal{SREN}$,[1] then we get a two-round malicious secure protocol in the OT correlations model for $\mathcal{SREN}$ that has statistical security. This give a "round-optimal" version of the corresponding information-theoretic result from [IPS08] making black-box use of the inner protocol. A second corollary is a two-round, statically secure malicious MPC protocol for computing log-depth arithmetic circuits. This protocol is secure in the presence of OLE + OT correlations[2] and additionally makes black-box use of the underlying field. This is obtained by instantiating the inner protocol with the protocol of Lin et al. [LLW20]. This settles an open problem from [LLW20] on constructing 2-round protocols over OLE with statistical security against malicious adversaries.

Towards concretely efficient 2-sided NISC. Another interesting use case for the above result is the 2-round, secure *two-party* protocol in which *both* parties get an output. This should be contrasted with the standard notion of non-interactive secure computation (NISC) [IKO$^+$11] that applies to one-sided functionalities. Note that this kind of *2-sided NISC* cannot be obtained by simply running two parallel instances of standard NISC, since even if we ignore parallel composition issues, there is no mechanism to enforce consistency between the inputs used in these instances (unless we rely on zero-knowledge proofs and make non-black-box use of cryptography). The only alternative black-box approach to 2-sided NISC over OT correlations we are aware of is via the protocol garbling technique that garbles the code of a malicious secure protocol and thus, has prohibitive computational and communication cost. Even in the 1-sided case, existing protocols from [IKO$^+$11, AMPR14, MR17, HIV17, CDI$^+$19] are heavily tailored to specific garbling techniques and do not make a black-box use of an underlying semi-honest protocol.

We note that techniques developed in the context of an "IPS-style compiler" in the two-round setting gives a new approach for constructing protocols for the 2-sided NISC problem. Specifically, if we use [IKP10, Pas12] as the outer protocol and use the simple two-sided version of Yao's protocol (using Boolean garbling in the OT correlations model) as the inner protocol, we obtain a 2-sided NISC protocol that is secure against malicious adversaries in the OT correlations model.[3] In Section 8.5, we suggest some optimizations to improve the concrete cost of this protocol.

[1]$\mathcal{SREN}$ denotes the class of functions that admit an efficient statistical randomized encoding.

[2]OT can be non-interactively reduced to OLE over fields of small characteristic. However, given that OT is typically cheaper to implement than OLE, a setup of OLE+OT is arguably comparable to OLE alone.

[3]As we noted before, for the case of constant number of parties, watchlist correlations reduces to standard OT correlations.

1.1.3 Black-box 5-round MPC in the plain model.

Our third and final result uses a specialized combination of the previous contributions to get "one round away" from settling the main open question about the round complexity of black-box MPC. Concretely, we get a 5-round MPC protocol that makes a black-box use of PKE with pseudorandom public keys (as in the first contribution), along with any 2-round OT protocol with "semi-malicious" security. The latter security requirement is a very mild strengthening of semi-honest security in the context of 2-round OT protocols, and is satisfied by most 2-round OT protocols from the literature.

1.2 Related Work

In this subsection, we give a brief overview of the two main approaches taken by the prior work in obtaining black-box MPC protocols in the plain model.

Coin tossing based approach. The main idea in this approach is to use a black-box simulatable coin tossing protocol to setup a CRS and then use black-box MPC protocols (such as [GIS18]) in the CRS model. Roughly, to generate the CRS, the idea is for each party to commit to a random string r_i and in a later step, for all parties to reveal their coins. To ensure that malicious parties cannot set their randomness as a function of that of other honest players, players should use a (concurrent) non-malleable commitment in the commit phase.

But the main bottleneck to obtaining such a coin tossing protocol is achieving simulatability. To achieve the simulation guarantee and allow a simulator to "force" the output of the coin toss to be a certain value[4], one would need to rely zero-knowledge protocols, which if applied naively make black-box use of cryptography. Even if one were able to achieve simulation-based guarantees via a specific protocol, one would need to tailor this to prove statements about construction of bounded concurrent non-malleable commitment w.r.t. commitment against synchronising adversaries, for which no round efficient black-box constructions exist. More specifically, [GLOV12] gives a black-box protocol but the number of rounds of this protocol is greater than 18 (the coin tossing requires at least two more rounds. [GPR16] gives a 3-round black-box construction of NMCom but is only secure in the standalone setting. The other round efficient constructions of concurrent NMCom [GRRV14, COSV16, COSV17, Khu17] make non-black use of cryptography.

IPS compiler based approach. The IPS compiler [IPS08] gives a black-box MPC protocol in the OT hybrid model. The main challenge in instantiating this approach in the plain model is in constructing a protocol that securely realizes the ideal OT functionality. In particular, we need a protocol that realizes the ideal OT functionality between every ordered pair of parties. [Wee10] gave a non-constant round black-box way to realize this and this was improved by [Goy11] who gave a constant round black-box protocol. The main component in the constant round protocol is again a constant round black-box bounded concurrent non-malleable commitment wrt replacement (which is weaker than the traditional definition of non-malleable commitment wrt commitment). Even if we rely on a three-round black-box version of such a non-malleable commitment from [GKP$^+$18], the OT protocol requires at least 12 rounds of communication. A straightforward way of combining this with the IPS approach incurs at least 4 more rounds.

[4]Note that this corresponds to the programmability requirement.

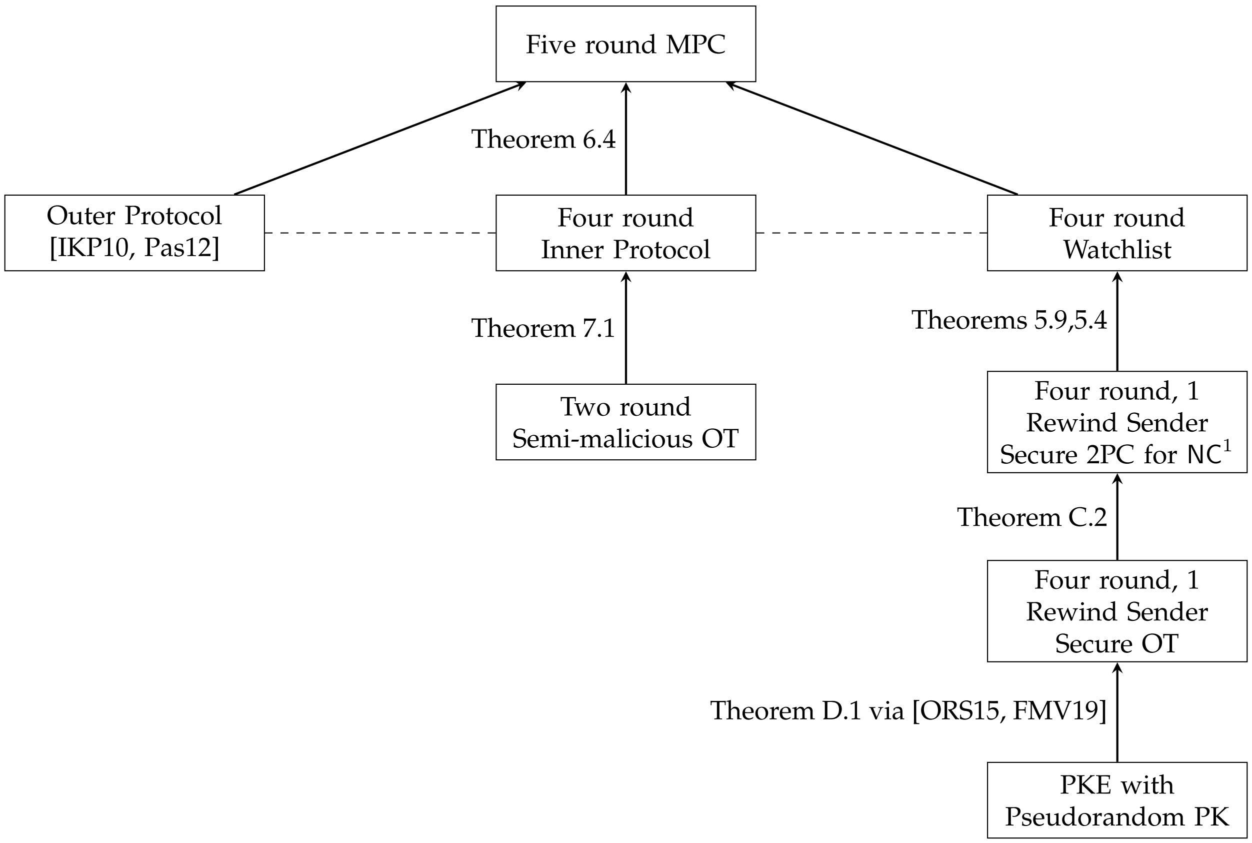
Five round MPC
Theorem 6.4
Outer Protocol [IKP10, Pas12]
Four round Inner Protocol
Four round Watchlist
Theorem 7.1
Two round Semi-malicious OT
Theorems 5.9,5.4
Four round, 1 Rewind Sender Secure 2PC for NC^1
Theorem C.2
Four round, 1 Rewind Sender Secure OT
Theorem D.1 via [ORS15, FMV19]
PKE with Pseudorandom PK

Figure 1: Overview of the Key Steps in our Black-Box Five-Round Protocol.

2 Technical Overview

In this section, we provide an overview of the key technical ideas underlying our protocols. In Figure 1, we give a high-level overview of the key steps involved in obtaining our five-round black-box MPC protocol.

2.1 Background: Black-box MPC in the OT Hybrid Model

We begin with a brief recap of the MPC-in-the-head approach of Ishai, Prabhakaran and Sahai [IPS08], which is the starting point for our work. Readers familiar with this paradigm may skip to Section 2.2.

We stress that [IPS08] (henceforth referred to as the IPS paradigm) does not focus on round complexity, or on obtaining constructions in the plain model. Nevertheless our work uses this approach as a meaningful starting point towards obtaining round efficient MPC in the plain model.

The IPS paradigm combines the following components to obtain an n-party MPC protocol, in the OT hybrid model, with security against malicious corruptions of all but one players.

- An "outer" MPC in the client-server model, with n clients and m servers. This protocol is secure against malicious corruption of $O(m)$ servers, and $n-1$ clients.
- An "inner" n-party MPC secure against semi-honest corruptions of upto $n-1$ players.

- A "watchlist" that enables parties to *check* each other and detect malicious behaviour.

An Outer Protocol with Minimal Interaction [IKP10, Pas12]. Before delving into details of the IPS technique, it will help to fix a special (minimal) interaction pattern for the outer protocol. The outer protocol is executed between n clients and m servers, and as discussed above, must be secure even if $n-1$ clients collude maliciously with a constant fraction of the servers (concretely, we will fix this fraction to be 1/3). We will assume that this protocol has the following interaction pattern:

- Every client $\mathcal{C}_i$ for $i \in [n]$ sends a single message $\phi_{i \to j}$ to each server $\mathcal{S}_j$ for $j \in [m]$.
- Each server $\mathcal{S}_j$ for $j \in [m]$ performs local computation on its view, i.e. computes $\mathcal{S}_j(\{\phi_{i \to j}\}_{i \in [n]})$ which outputs $\phi_{j \to i}$ for each client P_i, where $i \in [n]$.
- Each client recovers its output by performing local computations on its view $\{\phi_{j \to i}\}_{j \in [m]}$.

An information-theoretic protocol as above exists for "simple" functionalities (e.g., in the complexity class NC^1) [IKP10]. This can be extended to a similar protocol for general functionalities that makes black-box use of any PRG (see Section 3.5.3.5 of [Pas12]).

The IPS Paradigm [IPS08]. We briefly recall the IPS approach that combines an arbitrary inner and outer protocol to obtain an n-party black-box MPC protocol tolerating upto $(n-1)$ malicious corruptions. While the IPS technique works with arbitrary outer protocols, we will assume the minimal interaction pattern discussed above to simplify discussion and build towards our goal of round efficient black-box MPC.

Each party plays the role of a "client" in the outer protocol. At the same time, all the n parties use the inner protocol to *jointly* emulate the local computation of each of the m "servers" in the outer protocol. That is, parties run m instances of the inner protocol, where the j^{th} instance for $j \in [m]$ corresponds to a semi-honest secure computation of the j^{th} server's functionality $\mathcal{S}_j$.

Recall that the objective is to achieve security against upto $n-1$ *malicious corruptions*. Since the inner protocol only guarantees security against semi-honest corruptions, if *any* of the parties emulating the j^{th} server behave maliciously, the server $\mathcal{S}_j$ cannot be trusted and is essentially corrupted. Now recall that the outer protocol is only secure against (malicious) corruption of a $1/3^{rd}$ of the servers. Therefore, one must necessarily ensure that *all parties are semi-honest* in $2/3^{rd}$ of the m inner protocol sessions.

Such semi-honest behavior is enforced with the help of a sophisticated cut-and-choose mechanism, that is implemented via "watchlists". A watchlist enables every party P_i to *watch* (i.e., check) a private small subset $\mathbb{S}_i$ of inner protocol instances. Specifically, for every $j \in \mathbb{S}_i$, party P_i obtains the purported input and randomness used by *other parties* in instance j. Given these values, each P_i reconstructs protocol messages for all sessions $j \in \mathbb{S}_i$, and aborts if it detects an inconsistency. Importantly, the watchlist must reveal no information about other sessions.

The size of each $\mathbb{S}_i$ is carefully tuned to ensure that the secrets of sufficiently many instances remain hidden from all parties. This ensures that:

- *If an adversary deviates from semi-honest behaviour in more than* $1/3^{rd}$ *of the* m *inner protocol sessions*, then w.h.p., at least one of the deviating executions will be on the "watchlist" of (i.e., will be checked by) an honest party, causing the protocol to abort.
- *If the adversary deviates from semi-honest behaviour in less than* $1/3^{rd}$ *of the* m *inner protocol sessions*, then $2/3^{rd}$ of the outer protocol servers are uncorrupted. This makes it possible to rely on security of the outer protocol against upto $1/3^{rd}$ fraction of server corruptions.

The IPS approach realized watchlists in the OT hybrid model, i.e. assuming participants have access to ideal OTs. As a result, they obtained a black-box, constant round MPC protocol tolerating all-but-one corruptions in the OT hybrid model. As already discussed in the introduction, our key technical contributions are to obtain constructions of watchlists and round efficient MPC in the *plain model* based on black-box use of simple primitives.

2.2 The Watchlist Protocol

As a first step, we introduce novel techniques to build round-optimal watchlists in the plain model, that make black-box use of any public-key encryption (PKE) with pseudorandom public keys.

We begin by closely inspecting an ideal version of the watchlist functionality. For $i \in [n], j \in [n] \setminus \{i\}$, this functionality must enable party P_i to choose a random (secret) subset $\mathbb{S}_i \subseteq [m]$ of protocol executions of size k, and obtain *only* the purported input and randomness used by P_j in all executions in the set $\mathbb{S}_i$. At the same time all other inputs of P_j should remain hidden from P_i. A careful reader would have already observed that the watchlist functionality can be realized if we have access to ideal, pairwise k-out-of-m OT functionality between every pair of parties. We observe that the k-out-of-m OT is a one-sided functionality and hence, this can be realized if parties have pairwise access to independent copies of the ideal OT functionality [IPS08, IKO$^+$11]. We call this as simultaneous secure OT and would like to securely realize this ideal functionality in the plain model in the presence of *arbitrary malicious* corruptions.

A Starting Point. A natural first attempt is to just have each pair of parties simultaneously execute a two-party secure protocol computing the k-out-of-m OT functionality. Such a protocol can be realized based on black-box use of any public key encryption scheme with pseudorandom public keys [FMV19, ORS15].

Unfortunately, this *does not* securely emulate access to independent copies of the ideal OT functionality between pairs of participants, because this protocol satisfies only *stand-alone* security. It is easy to achieve OT that composes under parallel repetition with *fixed roles*, i.e. where many OT sessions are executed in parallel, and an adversary either corrupts multiple senders or multiple receivers but does *not* simultaneously corrupt (subsets of) senders and receivers. In particular, the stand-alone secure construction of OT from pseudorandom public keys in [FMV19] already achieves this notion of parallel composition.

But in the (more general) simultaneous setting, an adversarial party P_i^* participates in many OT sessions simultaneously, as sender in some sessions and receiver in others. This gives P_i^* the opportunity to generate its own (eg, sender) message in some OT session as a function of a message generated by an honest sender in a different OT session, thereby possibly making its own input depend on the input(s) of honest player(s). Clearly, this is disallowed by the ideal simultaneous OT functionality; but not prevented by standalone OT. Our first step towards addressing this vulnerability is to ensure that adversarial inputs are independent of the inputs of honest players.

As discussed in the introduction, we develop a novel approach to achieving such independence. In particular, we construct "non-malleable OT" that satisfies the following guarantees.

- **Receiver Security under Parallel Composition.** For every adversarial sender $\mathcal{A}^*$ that corrupts the OT sender (or resp., multiple senders in any parallel composition of the OT protocol), there exists a simulator that simulates the view of $\mathcal{A}^*$ with black-box access to (resp., copies of) the ideal OT functionality. This follows automatically from simulation-based security against malicious senders (resp., in the parallel composition setting) of the underlying two-party secure protocol $\Pi_{\mathcal{F}}$.

- **Non-Malleability.** Informally, here we consider a man-in-the-middle adversary MIM that acts as a receiver in a subset of OT sessions (that we refer to as "left" sessions) and as sender in a different subset of OT sessions (that we refer to as "right" sessions).

  We require the existence of a *simulator-extractor* $\mathsf{Sim\text{-}Ext}$, that given the inputs of all honest receivers (participating in all right sessions), is able to extract all the implicit inputs used by the MIM *in all its right sessions.* Crucially, $\mathsf{Sim\text{-}Ext}$ *does not* have access to the inputs of honest senders (participating in the left sessions).

  This is the key property that prevents an adversarial sender from "copying" the inputs of honest senders, or more generally, generating its inputs as a function of honest senders' inputs. Achieving this property will be a key technical focus of our work.

In what follows, we provide an overview of our construction of non-malleable OT. Then, in Section 2.2.2, we discuss why any non-malleable OT protocol satisfying these properties almost directly implies pairwise ideal OT functionality (or, simulataneous secure OT), and therefore also securely realizes watchlists.

2.2.1 Towards Non-Malleable OT

We take inspiration from recent works that use non-malleable codes (introduced in [DPW18]) to build cryptographic primitives like non-malleable commitments [GPR16] and non-malleable multi-prover interactive proofs [GJK15].

In more detail, we will build simultaneous secure OT by combining *parallel composable* two-party secure computation with (an) appropriate (variant of) split-state non-malleable codes. Such codes are specified by encoding and decoding algorithms $(\mathsf{Enc}, \mathsf{Dec})$. The encoding algorithm Enc is a randomized algorithm that encodes any message m into a codeword consisting of two parts or "states" (L, R), and the decoding algorithm Dec on input a codeword returns the underlying message. The security property is that for every pair of tampering functions (f, g) with no fixed points, the (distribution of) $\widetilde{m} \leftarrow \mathsf{Dec}(f(\mathsf{L}), g(\mathsf{R}))$, where $(\mathsf{L}, \mathsf{R}) \leftarrow \mathsf{Enc}(m)$, is independent of m. We now describe (a simplified variant of) our construction.

Our Construction For simplicity, we will focus on the special case of implementing simultaneous secure 1-*out-of-*2 *OT*, but our techniques immediately extend to the more general setting of k-out-of-m OT. To prevent obvious copying attacks, we will assign to each party a unique tag or identity.

Our construction of non-malleable OT simply involves *executing a secure two-party protocol* Π between a sender $\mathcal{S}$ and a receiver $\mathcal{R}$, for a special functionality $\mathcal{F}$. Before describing this functionality, we discuss the inputs of participants to this functionality.

The sender $\mathcal{S}$ with on input $(\mathsf{m}_0, \mathsf{m}_1)$ and tag encodes these messages using an appropriate split-state non-malleable code $(\mathsf{Enc}, \mathsf{Dec})$. Specifically, $\mathcal{S}$ computes $\mathsf{L}_0, \mathsf{R}_0 \leftarrow \mathsf{Enc}(\mathsf{m}_0 || \mathsf{tag})$ and $\mathsf{L}_1, \mathsf{R}_1 \leftarrow \mathsf{Enc}(\mathsf{m}_1 || \mathsf{tag})$. The receiver $\mathcal{R}$ obtains as input a choice bit $\mathsf{b} \in \{0, 1\}$, and samples uniformly random $c \in \{0, 1\}$. $\mathcal{S}$ and $\mathcal{R}$ then invoke a two-party secure protocol $\Pi_{\mathcal{F}}$ to compute functionality $\mathcal{F}$ described in Figure 2. In addition, $\mathcal{S}$ sends tag to $\mathcal{R}$.

Sender Inputs: $\mathsf{m}_0, \mathsf{L}_0, \mathsf{R}_0, \mathsf{m}_1, \mathsf{L}_1, \mathsf{R}_1, \mathsf{tag}$. **Receiver Inputs:** b, c.

The functionality $\mathcal{F}(\mathsf{m}_0, \mathsf{L}_0, \mathsf{R}_0, \mathsf{m}_1, \mathsf{L}_1, \mathsf{R}_1, \mathsf{b}, c, \mathsf{tag})$ is defined as follows.

1. If $\mathsf{Dec}(\mathsf{L}_0, \mathsf{R}_0) \neq (\mathsf{m}_0||\mathsf{tag})$ or $\mathsf{Dec}(\mathsf{L}_1, \mathsf{R}_1) \neq (\mathsf{m}_1||\mathsf{tag})$, output $\perp$.
2. If $c = 0$, output $(\mathsf{m}_b, \mathsf{L}_0, \mathsf{L}_1)$.
3. If $c = 1$, output $(\mathsf{m}_b, \mathsf{R}_0, \mathsf{R}_1)$.

Figure 2: The functionality $\mathcal{F}$

We note that the ideal functionality $\mathcal{F}$ reveals m_b to $\mathcal{R}$, and in addition, reveals either *only* $(\mathsf{L}_0, \mathsf{L}_1)$ or *only* $(\mathsf{R}_0, \mathsf{R}_1)$. Because any split-state non-malleable code is also a 2-out-of-2 secret sharing scheme [ADKO15], the shares L_{1-b} and R_{1-b} each statistically hide m_{1-b} from $\mathcal{R}$. It is also clear that protocol Π makes only black-box use of the underlying two-party computation protocol.

We show that (an appropriate k-out-of-m variant of) the protocol sketched above securely realizes non-malleable OT, even when Π itself is only parallel composable secure (but may be completely malleable).

Proving Sender Non-Malleability. For ease of exposition, let's consider a simpler man-in-the-middle adversary (MIM) that participates in a single left session as receiver, and a single right session as sender. We will also assume that the MIM never sends messages that cause an honest party to abort. Additionally, the underlying secure two-party protocol Π will be a round optimal (four round) protocol with sequential messages, and with a specific structure. Namely, it will require the receiver to commit to its input b in the first round of the protocol, and at the same time, it will be *delayed-input* w.r.t. receiver input c, which will be chosen by the receiver immediately before the third round begins. Finally, it will require the inputs $(\mathsf{m}_0, \mathsf{m}_1, \mathsf{L}_0, \mathsf{R}_0, \mathsf{L}_1, \mathsf{R}_1, \mathsf{tag})$ of the sender to be committed in the second round of the protocol, *before c is chosen by the receiver*.

First Attempt: An Alternate Extraction Mechanism. One possible way to extract sender inputs from the right execution, is to execute the simulator of the underlying two-party protocol Π. Unfortunately, this fails because Π is only parallel composable secure, and extracting from the right execution automatically reveals honest sender inputs from the left execution.

Instead, we will use the specific way that sender inputs are encoded to introduce an *alternate* extraction mechanism. Specifically, one could imagine rewinding the third and the fourth round message of Π, using inputs $c = 0$ and $c = 1$ on behalf of the honest receiver in the real and rewinding threads, respectively. By our assumption, the adversary is non-aborting. Therefore, we expect to obtain outputs $(\widetilde{\mathsf{L}}_0, \widetilde{\mathsf{L}}_1)$ and $(\widetilde{\mathsf{R}}_0, \widetilde{\mathsf{R}}_1)$ in the right session in the real and rewinding threads respectively. At this point, we can use the decoder of the non-malleable code to obtain $(\widetilde{\mathsf{m}}_0, \widetilde{\mathsf{m}}_1)$, which, by correctness of the two-party protocol, should correspond to the implicit inputs of the MIM in the right session.

It doesn't seem like this argument gives us much (yet): rewinding the MIM's third and fourth rounds would also end up rewinding the third and the fourth rounds of the left execution with (possibly different) inputs $\widetilde{c}, \widehat{c}$ used by the MIM in the main and rewinding threads. Thus, it may seem like we are back to square one: it may not be possible to hide the inputs of the honest sender in the presence of such rewinding.

Towards Resolving the Extraction Bottleneck: 1-Rewind Sender Security. To tackle this problem, our first step will be to require that Π satisfy a stronger security property: 1-*rewind sender security*. Roughly, this means that any adversarial receiver $\mathcal{R}^*$ that rewinds the honest sender one time in the third and fourth rounds, using (possibly different) inputs $\widetilde{c}, \widehat{c}$ in the main and rewinding threads, does not recover any information beyond the output of $\mathcal{F}$ on inputs $(\mathsf{m}_0, \mathsf{m}_1, \mathsf{L}_0, \mathsf{L}_1, \mathsf{R}_0, \mathsf{R}_1, \widetilde{\mathsf{b}}, \widetilde{c})$ and $(\mathsf{m}_0, \mathsf{m}_1, \mathsf{L}_0, \mathsf{L}_1, \mathsf{R}_0, \mathsf{R}_1, \widetilde{\mathsf{b}}, \widehat{c})$. We formalize this by requiring the existence of a specific type of simulator: this simulator generates a view for $\mathcal{R}^*$ in the main thread given only $(\mathsf{m}_{\widetilde{\mathsf{b}}}, \mathsf{L}_0, \mathsf{L}_1)$ and a view for $\mathcal{R}^*$ in the rewinding thread given only $(\mathsf{m}_{\widetilde{\mathsf{b}}}, \mathsf{R}_0, \mathsf{R}_1)$ (or vice-versa). Now, it may seem like this type of simulator may not be very meaningful, since the sum total of this information could essentially allow the receiver to recover $\mathsf{m}_{1-\widetilde{\mathsf{b}}}$ by combining $\mathsf{L}_{1-\widetilde{\mathsf{b}}}$ with $\mathsf{R}_{1-\widetilde{\mathsf{b}}}$.

However, the fact that $(\mathsf{L}_0, \mathsf{L}_1)$ and $(\mathsf{R}_0, \mathsf{R}_1)$ are made available in *separate* threads can be exploited argue that the MIM's input must be independent of $\mathsf{m}_{1-\widetilde{\mathsf{b}}}$, as we discuss next.

Alternative Simulation. Let us go back to our alternate extraction mechanism discussed earlier, where w.l.o.g. the third and fourth round messages of Π are rewound with (honest) receiver input set to $c = 0$ in the main and $c = 1$ in the rewinding thread, respectively. This means that in the main thread, the challenger obtains output $(\widetilde{\mathsf{L}}_0, \widetilde{\mathsf{L}}_1)$ in the right session. In the rewind thread, setting $c = 1$, the challenger obtains outputs $(\widetilde{\mathsf{R}}_0, \widetilde{\mathsf{R}}_1)$. Meanwhile the real and rewinding left executions will simulated using only $(\mathsf{m}_{\widetilde{\mathsf{b}}}, \mathsf{L}_0, \mathsf{L}_1)$ and $(\mathsf{m}_{\widetilde{\mathsf{b}}}, \mathsf{R}_0, \mathsf{R}_1)$ (or vice-versa) respectively, as described above. *This means that in the main thread, the* MIM *outputs* $(\widetilde{\mathsf{L}}_0, \widetilde{\mathsf{L}}_1)$ *as a function of only* $(\mathsf{m}_{\widetilde{\mathsf{b}}}, \mathsf{L}_0, \mathsf{L}_1)$*, and in the rewinding thread, the* MIM *outputs* $(\widetilde{\mathsf{R}}_0, \widetilde{\mathsf{R}}_1)$ *as a function of only* $(\mathsf{m}_{\widetilde{\mathsf{b}}}, \mathsf{R}_0, \mathsf{R}_1)$.[5]

We formalize this intuition to argue that the MIM's behaviour naturally gives rise to a split-state tampering function family. Here, one tampering function corresponds to the MIM's functionality in the main thread, and the other corresponds to the MIM's functionality in the rewinding thread. This allows us to rely on the non-malleability of the underlying encoding scheme to switch from generating $\mathsf{L}_{1-\widetilde{\mathsf{b}}}, \mathsf{R}_{1-\widetilde{\mathsf{b}}}$ as an encoding of $m_{1-\mathsf{b}}$, to generating it as an encoding of a dummy value.

This completes a simplified description of the main ideas in our protocol. We swept several details under the rug but point out one important detail below.

Many-many Non-malleability. Recall that we simplified things earlier, to focus on a setting where the MIM participates in a single left session as receiver and a single right session as sender. For our application to watchlists, we require security against adversaries that participate in *multiple* left and right sessions.

To achieve security in this setting, we will rely on *many-many* non-malleable codes (that are implied by one-many non-malleable codes [CGL16]) that achieve security in the presence of multiple tamperings of a single codeword [CGL16]. Moreover, in order to deal with adversaries that may abort arbitrarily, we will modify the functionality $\mathcal{F}$. Instead of encoding (m_0, m_1) a single time, the sender generates λ (where λ is the security parameter) fresh encodings $\{(\mathsf{L}^i_\mathsf{b}, \mathsf{R}^i_\mathsf{b})\}_{i\in[\lambda],\mathsf{b}\in\{0,1\}}$ of m_0 and m_1. The receiver picks λ choice bits $c_1, \ldots, c_\lambda$ instead of a single bit c. The functionality $\mathcal{F}$ checks if for every $i \in [\lambda]$, $\mathsf{b} \in \{0,1\}$, $\{(\mathsf{L}^i_\mathsf{b}, \mathsf{R}^i_\mathsf{b})\}_{i\in[\lambda],\mathsf{b}\in\{0,1\}}$ encode m_b. If the check fails, $\mathcal{F}$ outputs $\perp$. If it passes, then for every $i \in [\lambda]$, it outputs $(\mathsf{L}^i_0, \mathsf{L}^i_1)$ if $c_i = 0$ and otherwise, outputs $(\mathsf{R}^i_0, \mathsf{R}^i_1)$.

[5]Actually, the MIM may also output $(\widetilde{\mathsf{L}}_0, \widetilde{\mathsf{L}}_1)$ as a function of only $(\mathsf{m}_{\widetilde{\mathsf{b}}}, \mathsf{R}_0, \mathsf{R}_1)$, and $(\widetilde{\mathsf{R}}_0, \widetilde{\mathsf{R}}_1)$ as a function of only $(\mathsf{m}_{\widetilde{\mathsf{b}}}, \mathsf{L}_0, \mathsf{L}_1)$. We use codes satisfying an additional *symmetric decoding* property to account for this case.

This helps ensure that for every adversary MIM that completes a main thread (without aborting) given honest receiver input $c = c_1, \ldots, c_\lambda$, there is (w.h.p.) a rewinding thread with a different choice $c' = c'_1, \ldots, c'_\lambda$ of honest receiver input, that is also completed by the MIM. We then rely on any index i for which $c_i \neq c'_i$ to carry out the argument described above. Additional details of our non-malleable OT protocol can be found in Section 5.1.

2.2.2 From Non-Malleable OT to Watchlists

We note that that our OT protocol, as described above, prohibits an adversarial sender from generating its generating its inputs as a function of honest senders' inputs.

One could ask for an even stronger property, requiring the inputs of adversarial receivers to be independent of the honest receivers' inputs. At first glance, this stronger property appears to be necessary, since pairwise access to ideal OTs would actually enforce that all adversarial receiver inputs are independent of the inputs of honest receivers.

But upon taking a closer look, we realize that non-malleable OT as described in the previous section actually suffices to construct watchlists with security in the real/ideal paradigm. Intuitively, this is because the outputs of honest parties are affected only by the inputs of the adversarial senders, and are unaffected by the inputs of adversarial receivers. In other words, even if adversarial receivers manage to have their inputs depend on the inputs of the honest receivers, this cannot affect the joint distribution of the view of adversary and the outputs of honest parties in the ideal world.

In Section 5, we show that non-malleable OT implies watchlists, with security according to the real/ideal definition. We achieve that via a careful hybrid analysis. First, we rely on receiver security under parallel composition – and replace honest receivers' inputs with dummy inputs while extracting inputs of all senders. In the next hybrid step, we rely on *sender non-malleability* to replace the inputs of all honest senders with fake inputs, while arguing that the distribution of inputs extracted from malicious senders (and therefore also the distribution of honest party outputs) remains indistinguishable from the real protocol. This completes a high-level overview of our watchlist construction, and the detailed proof can be found in Section 5.2. The only missing ingredient in our description is the 1-rewind sender secure protocol, which we describe next.

2.2.3 Constructing a 1-Rewind Sender Secure Protocol

In our actual construction of non-malleable OT, the receiver inputs $(c_1, \ldots c_\lambda)$ do not need to remain hidden from a corrupted sender. In particular, all we need is for the protocol to allow for delayed *function* selection, where the function to be computed (defined by $c_1, \ldots, c_\lambda$) is selected by the receiver in the third round. Given this, the 1-rewinding security property translates to requiring that any corrupt receiver which rewinds the third and the fourth round messages of the sender by providing (possibly) different functions learns nothing beyond the output of these two functions on sender and receiver inputs that were fixed in the first two rounds.

We will design such a 2-party protocol for $\mathsf{NC1}$ circuits[6] by relying on a different variant [IKOS07, IPS08, IKO$^+$11] of the IPS paradigm. Specifically, we will use the same 2-client m-server outer protocol [Pas12] that was discussed at the beginning of the overview, and combine it an inner protocol that is based a variant of Yao's garbled circuits [Yao86]. Yao's protocol also allows for the garbled circuits to be generated in the final round, which immediately gives us the delayed function selection property. Importantly, since we only care about parallel composable security in the resulting two-party protocol, parallel composable but possibly malleable 1-rewind secure OT will suffice

[6]We show in Section 5.1 that 1-rewind secure 2PC for $\mathsf{NC1}$ circuits suffices to obtain non-malleable OT.

to implement watchlists in this setting. We slightly generalize the works of [ORS15, FMV19] to obtain a maliciously secure OT that satisfies 1-rewind sender security. We refer the reader to Section C for the details of constructing the secure computation protocol and to Appendix D for the construction of a 1-rewind sender secure OT protocol.

2.2.4 Immediate Application: Black-box Simultaneous Two-Party Computation

Plugging the resulting simultaneous OT protocol in place of ideal OT, into the non-interactive two-party secure black-box computation protocol of [IKO$^+$11], yields a round optimal two-party *simultaneous* secure computation, from black-box use of any PKE with pseudorandom public keys. Next, we discuss our key application: round-efficient black-box MPC.

2.3 Two-Round Protocol in the OT Correlations Model

We now give details of our two-round MPC protocol in the OT correlations model that makes black-box use of any two-round protocol that satisfies an augmented notion of semi-honest security (described in detail below). As mentioned before, this result relies on the IPS paradigm (recalled in Section 2.1) and it will serve as a useful stepping stone towards our final five-round protocol in the plain model.

One-Round Watchlists in the OT Correlations Model. Let us assume for simplicity that the parties have access to a k-out-of-m OT correlations functionality. In the main body, we will use a 1-out-of-k OT correlations functionality instead. In the OT correlations set-up phase, every pair of parties invoke the OT correlations functionality acting as the sender and the receiver respectively. The sender receives m uniformly chosen random strings and the receiver obtains a subset of these strings of size k. Given such correlations, it is easy to implement a single round watchlist protocol. Specifically, the randomly sampled sender inputs serve as secret keys/one-time pads that can be used to encrypt the actual sender inputs to the watchlist functionality. Given these ciphertexts, the receiver can use the keys obtained in the setup phase to recover the sender messages corresponding to its watched executions.

Outline. The main idea in our two-round protocol is to parallelize the single round watchlist protocol with the first round of the inner protocol. Before sending the second round messages of the inner protocol, each party verifies whether the input and randomness used by all parties in its watched executions are consistent with the first round message. If it detects any inconsistency, it aborts. Otherwise, it sends the second round message.

At first sight, it may appear that the above compiler can work with any two-round semi-honest protocol where the adversarial parties can use an arbitrary random tape (a.k.a. semi-malicious adversaries). However, if we look closely into the security proof, we need the two-round protocol to satisfy an additional property, namely, first round equivocality. To see why this is needed, consider an adversary that behaves maliciously in only a small number of inner protocol executions. Recall that in this case, security comes from the guarantees of the outer MPC protocol. However, if an adversary deviates in a very small number of executions, it is possible that the executions in which the adversary cheated were not watched or detected by any honest party. Therefore, the adversary can deviate and even potentially recover the input used by honest parties in these executions, and it may, in fact, be hard to efficiently compute which executions these were, until the end of the first round. Not knowing where the adversary could have cheated makes it tricky to invoke the inner MPC simulator, which is secure only in the presence of semi-malicious behavior.

We address this issue by requiring the inner protocol to admit a simulator that generates the messages for the first round of the inner protocol, without knowing whether the adversary cheated or not in these rounds. Before the last round, the simulator learns which sessions the adversary cheated in, and obtains honest party inputs for those sessions. At this point, the simulator equivocates the transcript generated so far to make it appear as if the honest party's input had been used all along. In summary, we require that the inner protocol satisfy the following properties:

- **Security in the Presence of Semi-Malicious Behavior.** The view of an adversary that behaves semi-maliciously in the first round can be simulated given the function's output.
- **Equivocality.** If an adversary *did not behave* semi-maliciously in the first round, then given inputs of honest parties at the end of the first round, the simulator generates a final message of the protocol corresponding to these inputs, that is indistinguishable from the real distribution.

We call a MPC protocol that satisfies these two properties as augmented semi-honest secure. In the multiparty setting, we note that the protocol given in [GIS18] in the OT correlations model satisfies both these properties. Combining it with our four round watchlist protocol (which can also be used to generate the OT correlations required by the [GIS18] protocol), this gives a six round protocol in the plain model that makes black-box use of public-key encryption with pseudorandom public keys. We refer the reader to Section 1.1 for details on other useful instantiations of the inner protocol in the two-party case.

2.4 Fully Black-Box MPC in Five Rounds

The key reason why the above approach leads to a six round protocol is that the first round of the inner [GIS18] protocol cannot be sent until the OT correlations are generated, which in turn requires four rounds.

We now propose an optimization that yields a 5-round fully black-box MPC protocol. Specifically, we build a special 4-round inner protocol that can be *parallelized* with our watchlists. At a high level, our inner protocol has the following structure: the first two rounds are used to generate appropriate OT correlations, and in the next two rounds, we run the [GIS18] round collapsing compiler on an information-theoretic protocol in the OT correlations model. This leads to a 4 round protocol, and we show that the first 3 rounds of this protocol can be parallelized with the last 3 rounds of our watchlist protocol.

A Subtle Issue. Note that our watchlist protocols allow parties to 'check' each other only at the end of round 4. In particular, a party may behave maliciously in the first three rounds, and this behavior will only be detected at the end of round 4, at which point it may be too late and the adversary might be able to completely break the inner protocol. Therefore, in order to be able to parallelize watchlists with an inner protocol, we must ensure that the inner protocol does not leak information in its initial rounds, even in the presence of malicious behavior. In more detail, we will need this protocol to satisfy an additional property, namely, robustness [ACJ17].

We say that an r-round protocol is robust if the honest party inputs remain hidden from an adversary that behaves maliciously in the first $(r-1)$-rounds. Thus, if we parallelize the first three rounds of the inner protocol with the last three rounds of the watchlist protocol, then robustness property guarantees that even if the adversary behaved maliciously in many executions, it cannot learn any information about the honest party's input at the end of the fourth round. But in this case, the watchlists guarantee that such an adversary will be caught with overwhelming probability and the honest parties will abort before sending the last round message of the inner protocol. On the other hand, if the adversary cheated only in a small number of executions, we require the inner protocol to additionally satisfy equivocality property as explained in Section 2.3. In summary, we require that the four-round inner protocol satisfy the following properties:

- **Robustness against Malicious Behavior.** Honest party inputs remain hidden from an adversary that arbitrarily (and maliciously) deviates from the protocol in the first 3 rounds.
- **Security in the Presence of Semi-Malicious Behavior.** The view of an adversary that behaves semi-maliciously in the first 3 rounds can be simulated given the function's output.
- **Equivocality.** If an adversary *did not behave* semi-maliciously in the first 3 rounds, then given inputs of honest parties at the end of round 3, the simulator generates a final message of the protocol corresponding to these inputs, that is indistinguishable from the real distribution.

We show that an instantiation of the four-round version of the [GIS18] protocol using a special two-round OT protocol that allows a simulator to extract the sender inputs satisfies all these properties. We give a construction of such an OT protocol based on any two-round semi-malicious oblivious transfer in Appendix B. We refer the reader to Section 7 for details of constructing the robust, equivocal MPC protocol.

Putting Things Together. To construct a five-round MPC protocol for general functions, we instantiate the IPS compiler with [IKP10, Pas12] as the outer protocol and use the above 4-round robust, equivocal MPC as the inner protocol. We implement the watchlist functionality using four-round protocol discussed earlier. We additionally parallelize the first 3 rounds of the inner protocol with the second, third and fourth rounds of the watchlist protocol. At the end of the fourth round, each party obtains the input and the randomness used by every other party corresponding to its watched executions. The party then checks if the inner-protocol messages obtained in those executions are consistent with the input and randomness obtained from the watchlist protocol. If any deviation is detected, then this party aborts. On the other hand, if all the watched executions are correct, then the party sends the final round message of the inner protocol. We refer the reader to Section 6 for additional details on how to prove security of the five-round MPC protocol.

2.5 Barriers to 4 Round Black-Box MPC

The works of [KO04, GMPP16] proved a 4 round lower bound for any MPC with black-box simulation. Their bound also applies to protocols that make non-black-box use of underlying primitives. A sequence of recent works [GMPP16, ACJ17, BGJ$^+$18, CCG$^+$20] obtained protocols that match this lower bound, while making non-black-box use of cryptography. On the other hand, the situation in the fully black-box setting has been relatively bleak. In particular, the best known prior round complexity here can be attributed to Goyal [Goy11], optimizations of which result in a protocol with at least 15 rounds. While our techniques help reduce this to 5 rounds, we encounter barriers when shaving off one extra round to obtain a round optimal construction.

All k round MPC protocols admitting a black-box simulator, require the simulator to extract the inputs of (possibly malicious players) by the end of the $(k-1)^{th}$ round, query the ideal functionality on these inputs, and obtain an output. The simulator is then required to "force" this output into the view of the adversary, by the end of the k^{th} round. It turns out that in our setting, such extraction must be accomplished by extracting sender inputs from watchlists. Furthermore, watchlists satisfying real-ideal security require at least 4 rounds, and allow extraction of sender inputs only by the end of the 4^{th} round. This is because they imply maliciously secure oblivious transfer with similar properties, which is known to require at least 4 rounds. As such, an extra round is required after the watchlist phase to allow the simulator to force an output, once it has extracted the adversary's input.

Previously, overcoming this barrier has involved developing techniques to simultaneously extract from the adversary and force an output. When making non-black-box use of cryptography, this has required the use of sub-exponential security [ACJ17] or special, complex primitives and highly non-black-box mechanisms such as conditional disclosure of secrets [BGJ+18, CCG+20]. We leave the task of building upon our conceptual framework by redesigning primitives and techniques especially tailored to the 4 round setting, as an interesting question for future work.

3 Preliminaries and Definitions

Let λ denote the security parameter. A function $\mu(\cdot) : \mathbb{N} \to \mathbb{R}^+$ is said to be negligible if for any polynomial $\mathsf{poly}(\cdot)$ there exists λ_0 such that for all $\lambda > \lambda_0$, we have $\mu(\lambda) < \frac{1}{\mathsf{poly}(\lambda)}$. We will use $\mathsf{negl}(\cdot)$ to denote an unspecified negligible function and $\mathsf{poly}(\cdot)$ to denote an unspecified polynomial function.

For a probabilistic algorithm A, we denote $A(x; r)$ to be the output of A on input x with the content of the random tape being r. When r is omitted, $A(x)$ denotes a distribution. For a finite set S, we denote $x \leftarrow S$ as the process of sampling x uniformly from the set S. We will use PPT to denote Probabilistic Polynomial Time algorithm.

We give the definition of Secure Multiparty Computation in Appendix A.

3.1 Two-Round Semi-Malicious Secure Oblivious Transfer

Syntax. Let $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ be a two-round oblivious transfer with the following syntax. OT_1 takes the security parameter 1^λ and the receiver's choice bit and outputs the first round message otm_1 along with some secret state ω. OT_2 takes otm_1 and the two sender inputs m_0 and m_1 and outputs otm_2. OT_3 takes otm_2 and (b, ω) and outputs m_b. We say that the OT protocol is secure against semi-malicious adversaries if it satisfies the following properties.

- **Correctness:** For every input b of the receiver and m_0, m_1 of the sender:

$$\Pr[\mathsf{OT}_3(\mathsf{otm}_2, (b, \omega) = m_b] = 1$$

  where $(\mathsf{otm}_1, \omega) \leftarrow \mathsf{OT}_1(1^\lambda, b)$ and $\mathsf{otm}_2 \leftarrow \mathsf{OT}_2(\mathsf{otm}_1, m_0, m_1)$.

- **Receiver Security.**

$$\{\mathsf{otm}_1 : (\mathsf{otm}_1, \omega) \leftarrow \mathsf{OT}_1(1^\lambda, 1)\} \approx_c \{\mathsf{otm}_1 : (\mathsf{otm}_1, \omega) \leftarrow \mathsf{OT}_1(1^\lambda, 0)\}$$

- **Sender Privacy:** For any input m_0, m_1 of the sender and any bit b and a string $r \in \{0,1\}^*$:

$$\{b, r, \mathsf{otm}_1 := \mathsf{OT}_1(1^\lambda, b; r), \mathsf{OT}_2(\mathsf{otm}_1, m_0, m_1)\} \approx_c \{b, r, \mathsf{otm}_1 := \mathsf{OT}_1(1^\lambda, b; r), \mathsf{OT}_2(\mathsf{otm}_1, m_b, m_b)\}$$

We note that the constructions of two-round oblivious transfer with statistical sender privacy [NP01, AIR01, Kal05, HK12, BD18] satisfies the above definition.

3.2 Non-malleable Codes

We will use non-malleable codes in the split-state model that are one-many secure and satisfy a special augmented non-malleability [AAG$^+$16] property, as discussed below.

Definition 3.1 (One-many augmented split-state non-malleable codes). *Fix any polynomials $\ell(\cdot), p(\cdot)$. An $\ell(\cdot)$-augmented non-malleable code with error $\epsilon(\cdot)$ for messages $m \in \{0,1\}^{p(\lambda)}$ consists of algorithms* $\mathsf{NM.Code}, \mathsf{NM.Decode}$ *where* $\mathsf{NM.Code}(m) \rightarrow (L, R)$ *where $L \in \mathcal{L}$ and $R \in \mathcal{R}$ (we will assume that $\mathcal{L} = \mathcal{R}$) such that for every $m \in \{0,1\}^{p(\lambda)}$,*

$$\mathsf{NM.Decode}(\mathsf{NM.Code}(m)) = m$$

and for every set of functions $f = (f_1, f_2, \ldots f_{\ell(\lambda)}), g = (g_1, g_2, \ldots g_{\ell(\lambda)})$ there exists a random variable $\mathcal{D}_{f,g}$ on $\mathcal{R} \times \{\{0,1\}^{p(\lambda)} \cup \mathsf{same}^*\}^{\ell(\lambda)}$ which is independent of the randomness in $\mathsf{NM.Code}$ *such that for all messages $m \in \{0,1\}^{p(\lambda)}$ it holds that*

$$\left|\left(\mathsf{R}, \{\mathsf{NM.Decode}\big(f_i(\mathsf{L}), g_i(\mathsf{R})\}_{i \in [\ell(\lambda)]}\big) \middle| (\mathsf{L}, \mathsf{R} \leftarrow \mathsf{NM.Code}(m))\right), (\mathsf{replace}(\mathcal{D}_{f,g}, m))\right| \leq \epsilon(\lambda) \text{ and}$$

$$\left|\left(\mathsf{R}, \{\mathsf{NM.Decode}\big(g_i(\mathsf{R}), f_i(\mathsf{L})\}_{i \in [\ell(\lambda)]}\big) \middle| (\mathsf{L}, \mathsf{R} \leftarrow \mathsf{NM.Code}(m))\right), (\mathsf{replace}(\mathcal{D}_{f,g}, m))\right| \leq \epsilon(\lambda)$$

where the function $\mathsf{replace} : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}$ *replaces all occurrences of* same^* *in its first input with its second input, and outputs the result.*

It was shown in [GKP$^+$18, GSZ20, ADN$^+$19] that the CGL one-many non-malleable codes constructed in [CGL16] are also one-many *augmented* non-malleable codes. But we point out that in this definition, we also require messages obtained by decoding the tampered codewords with *left and right shares interchanged* to be unrelated with the original message. It is easy to see that this property is satisfied by any non-malleable code with symmetric decoding (i.e. where $\mathsf{NMDec}(\mathsf{L}, \mathsf{R}) = \mathsf{NMDec}(\mathsf{R}, \mathsf{L})$). This property can simply be achieved, as observed in [GJK15], by modifying any split-state code to attach a special symbol "ℓ" to the left part of the codeword, and a special symbol "r" to the right part of the codeword. This yields the following imported lemma:

Lemma 3.2. *(Imported.) [GJK15, GKP$^+$18] For every polynomial $\ell(\cdot)$, there exists a polynomial $q(\cdot)$ such that for every $\lambda \in \mathbb{N}$, there exists an explicit ℓ-augmented, split-state non-malleable code satisfying Definition 3.1 with efficient encoding and decoding algorithms with code length $q(\lambda)$, rate $q(\lambda)^{-\Omega(1)}$ and error $2^{-q(\lambda)^{\Omega(1)}}$.*

3.3 Rewinding Secure Extractable Commitment

Definition 3.3. *An extractable commitment* $\mathsf{ECom} = (\mathsf{ECom}_1, \mathsf{ECom}_2, \mathsf{ECom}_3, \mathsf{ECom}_{\mathsf{Valid}})$ *is a three round protocol between two parties - a committer $\mathcal{C}$ with input $m \in \{0,1\}^\lambda$ and receiver $\mathcal{R}$ that proceeds as below.*

1. *$\mathcal{C}$ computes and sends $e_1 = \mathsf{ECom}_1(m; r_{\mathcal{C}})$ for uniform randomness r_c.*
2. *$\mathcal{R}$ computes and sends $e_2 = \mathsf{ECom}_2(e_1; r_{\mathcal{R}})$.*
3. *$\mathcal{C}$ computes and sends $e_3 = \mathsf{ECom}_3(e_2, m; r_{\mathcal{C}})$.*
4. *$\mathcal{R}$ outputs* $(\mathsf{accept/reject}) \leftarrow \mathsf{ECom}_{\mathsf{Out}}(e_1, e_2, e_3)$.

We require this commitment to satisfy the following properties:

- **1-Rewind-Security.** *For every non-uniform PPT malicious receiver $\mathcal{R}^*$ and every pair of messages $m_0, m_1\{0,1\}^\lambda$, the experiments $\mathsf{Exp}_{m_0,\mathcal{R}^*}$ $\mathsf{Exp}_{m_1,\mathcal{R}^*}$ are indistinguishable, where $\mathsf{Exp}_{m_b,\mathcal{R}^*}$ is defined as follows.*

  - *Run $\mathsf{ECom}_1(m_b; r_c)$ with uniform randomness r_c, and denote its output by (ec_1, σ), where σ is the committer's secret state, and ec_1 is the message to be sent to the receiver.*
  - *Run $\mathcal{R}^*(\mathsf{ec}_1)$, and obtain $\mathsf{ec}_2^0, \mathsf{ec}_2^1$.*
  - *For each $i \in \{0,1\}$, run $\mathcal{C}(\sigma, \mathsf{ec}_2^i)$ to obtain ec_3^i. Send the resulting messages $(\mathsf{ec}_3^0, \mathsf{ec}_3^1)$ to $\mathcal{R}^*$. The output of this experiment is the view of $\mathcal{R}^*$.*

- **(Over-)Extraction.** *There exists a PPT extractor $\mathsf{ECom}_{\mathsf{Ext}}$ that given oracle access to any (non-uniform) PPT malicious committer $\mathcal{C}^*$ outputs a pair (τ^*, σ^*) such that:*

  - *τ^* is identically distributed to the view of $\mathcal{C}^*$ (when interacting with an honest receiver) in the commitment phase, and*
  - *The probability that τ^* is an accepting transcript and $\sigma^* = \perp$ is negligible. Moreover, the probability that τ^* can be opened to to a value different than σ^* is negligible.*

Theorem 3.4. *[PRS02] Rewind-secure extractable commitments satisfying Definition 3.3 can be obtained based on black-box use of any non-interactive commitment scheme.*

3.4 Low-Depth Proofs

We will describe how any computation performed by a family of polynomial sized ciruits can be transformed into a proof that is verifiable by a family of circuits in $\mathsf{NC1}$. Let R be an efficiently computable binary relation. Let L be the language consisting of statements in R, i.e. for which $R(x) = 1$.

Definition 3.5 (Low-Depth Non-Interactive Proofs). *A low-depth non-interactive proof with perfect completeness and soundness for a relation R consists of an (efficient) prover P and a verifier V that satisfy:*

- **Perfect completeness.** *A proof system is perfectly complete if an honest provers can always convince an honest verifier. For all $x \in L$ we have*

$$\Pr[V(\pi) = 1 | \pi \leftarrow P(x)] = 1$$

- **Perfect soundness.** *A proof system is perfectly sound if it is infeasible to convince an honest verifier when the statement is false. For all $x \notin L$ and all (even unbounded) adversaries $\mathcal{A}$ we have*

$$Pr[V(x, \pi) = 1 | \pi \leftarrow \mathcal{A}(x)] = 0.$$

- **Low Depth.** *The verifier V can be implemented in* $\mathsf{NC1}$.

We outline a simple construction of a low-depth non-interactive proof, borrowed from [GGH$^+$13]. The prover P executes the verification circuit on x and generates the proof as the sequential concatenation (in some specified order) of the bit values assigned to the individual wires of the circuit computing R. The verifier V proceeds by checking consistency of the values assigned to the internal wires of the circuit for each gate. In particular for each gate in the verification circuit the verifier checks if the wire vales provided in the proof represent a correct evaluation of the gate.

Since the verification corresponding to each gate can be done independent of every other gate and in constant depth, we have that V itself is constant depth (with unbounded fan-in).

Looking ahead, our construction of non-malleable OT makes use of a (malleable) two-party computation protocol for $\mathsf{NC1}$ that must verify validity of a non-malleable code. We rely on low-depth proofs to ensure that the two-party computation protocol only performs $\mathsf{NC1}$ computations.

4 Definitions

4.1 Multi-Party Simultaneous OT

The (multi-party) simultaneous OT functionality is an n-party functionality that implements simultaneous (or parallel) α-out-of-β OT between n parties. Informally, this functionality consists of $n \cdot (n-1)$ instances of α-out-of-β OT. For every $i \in [n], j \in [n], j \neq i$, a secure OT is implemented between the pair (P_i, P_j) where P_i is sender and P_j is receiver.

Formally, we define the ideal (multi-party) simultaneous OT functionality $\mathcal{F}_{\mathsf{OT}}^{n\cdot(n-1)}$: this consists of $n(n-1)$ independent instances of α-out-of-β ideal OT, one for each ordered pair $(i, j) \in [n] \times [n]$ such that $i \neq j$. The (i, j)-th OT instance obtains input $x_{i,j}$ from sender P_i and input $y_{j,i}$ from receiver P_j. It parses $x_{i,j}$ as a set of β strings $(m_1, \ldots, m_\beta)$, and parses $y_{i,j}$ as a subset K of $[\beta]$ such that $|K| = \alpha$. It outputs $\{m_i\}_{i\in K}$ to P_j and outputs $\perp$ to P_i.

A (multi-party) simultaneous OT protocol is one that securely realizes the (multi-party) simultaneous OT functionality against malicious adversaries, according to Definition 4.1, as described below.

Definition 4.1 (n-Party (α, β) (Multi-party) Simultaneous OT Protocol)**.** *The (multi-party) simultaneous OT protocol* mpOT *is defined by a tuple of algorithms* $(\mathsf{mpOT}_1, \mathsf{mpOT}_2, \mathsf{mpOT}_3, \mathsf{mpOT}_4, \mathsf{out}_{\mathsf{mpOT}})$. *For each round $r \in [4]$, the i-th party in the protocol runs mpOT_r on 1^λ, the index i, the private input $\{x_{i,j}, y_{i,j}\}_{i\neq j}$ and the transcript of the protocol in the first $(r-1)$ rounds to obtain mpOT_r^i. It sends mpOT_r^i to every other party via a broadcast channel. We use $\mathsf{mpOT}(r)$ to denote the transcript of the protocol* mpOT *in the first r rounds. The output computing function $\mathsf{out}_{\mathsf{mpOT}}$ takes the index i of a party, its private input and its random tape and the transcript $\mathsf{mpOT}(4)$ and generates the output out_i. The protocol is required to satisfy:*

- ***Correctness:*** *For every choice of inputs $\{x_{i,j}, y_{i,j}\}_{i\in[n],j\neq i}$ for parties $\{P_i\}_{i\in[n]}$, we have:*

$$\Pr[(\mathsf{out}_1, \ldots, \mathsf{out}_n) = \mathcal{F}_{\mathsf{OT}}^{n\cdot(n-1)}(\{x_{i,j}, y_{i,j}\}_{i\in[n],j\neq i})] = 1$$

  where out_i denotes the output obtained by the i^{th} party from the mpOT *protocol when executed on the the private input and random tape of P_i.*

- ***Security:*** *Let $\mathcal{A}$ be an adversary corrupting an arbitrary subset of parties indexed by I, that obtains auxiliary input z. Let $\{\mathsf{REAL}_{\mathsf{mpOT},\mathcal{A}(z),I}(1^\lambda)\}_{\lambda\in\mathbb{N},z,\{x_{i,j},y_{i,j}\}_{i\in[n]\setminus I,j\neq i}}$ denote the joint distribution of the view of the adversary and the outputs of honest parties in an execution of the protocol with honest party inputs set to $\{x_{i,j}, y_{i,j}\}_{i\in[n]\setminus I,j\neq i}$.*

  We require the existence of an expected polynomial time simulator $\mathsf{Sim}_{\mathsf{mpOT}}$ that with black-box access to $\mathcal{A}$ interacts with the ideal $\mathcal{F}_{\mathsf{OT}}^{n\cdot(n-1)}$ functionality described above, and outputs an ideal view $\{\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{OT}}^{n\cdot(n-1)},\mathcal{A}(z),I}(1^\lambda)\}_{\lambda\in\mathbb{N},z,\{x_{i,j},y_{i,j}\}_{i\in[n]\setminus I,j\neq i}}$. The interaction between the simulator and the *ideal functionality is according to the standard model of secure computation*[7]*, with the following difference. The $\mathcal{F}_{\mathsf{OT}}^{n\cdot(n-1)}$ functionality does not wait to receive inputs for all $n\cdot(n-1)$ OT sessions before generating outputs, but generates outputs for the adversary for each session "on-the-fly", i.e. immediately after obtaining the inputs of both players. After that, in each session individually, it either aborts or sends the output to uncorrupted parties, depending on an instruction from the adversary for this session. We require that:*

$$\{\mathsf{REAL}_{\mathsf{mpOT},\mathcal{A}(z),I}(1^\lambda)\}_{\lambda\in\mathbb{N},z,\{x_{i,j},y_{i,j}\}_{i\in[n]\setminus I,j\neq i}} \approx_c \{\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{OT}}^{n\cdot(n-1)},\mathcal{A}(z),I}(1^\lambda)\}_{\lambda\in\mathbb{N},z,\{x_{i,j},y_{i,j}\}_{i\in[n]\setminus I,j\neq i}}$$

Remark 4.2 (Watchlist Protocol)**.** *The watchlist protocol is defined identically to the* mpOT *protocol (Definition 4.1).*

4.2 1-Rewind Sender-Secure Two-Party Computation

Let us consider a protocol Π between two parties, namely, the sender $\mathcal{S}$ and the receiver $\mathcal{R}$. The sender holds a private input $x_{\mathcal{S}}$ and the receiver holds a private input $x_{\mathcal{R}}$ and they wish to compute some function of their private inputs securely with the receiver obtains the output of the function. We want this protocol to satisfy:

- (Delayed-function selection) The function to be securely computed is only decided in the third round by the receiver $\mathcal{R}$. That is, the third round message contains the explicit description of the function f to be computed and the first two messages depend only on the size of the function.
- (1-Rewinding Security) Any malicious receiver that rewinds the third and fourth rounds of the protocol once (by possibly giving different functions f_0, f_1) cannot learn anything about the sender's inputs except the output on these two functions.

The syntax of the protocol and the two properties are formalized below.

Syntax. The special two party protocol Π is given by a tuple of algorithms $(\Pi_1, \Pi_2, \Pi_3, \Pi_4, \mathsf{out}_\Pi)$. Π_1 and Π_3 are the next message functions run by the receiver $\mathcal{R}$ and Π_2 and Π_4 are the next message functions run by the sender $\mathcal{S}$. At the end of the protocol, $\mathcal{R}$ runs out_Π on the transcript, its input and the random tape to get the output of the protocol. We use π_r to denote the message sent in the protocol Π in round r for every $r \in [4]$.

Definition 4.3. *Let $\Pi = (\Pi_1, \Pi_2, \Pi_3, \Pi_4, \mathsf{out}_\Pi)$ be a 4-round protocol between a receiver $\mathcal{R}$ and a sender $\mathcal{S}$ with the receiver computing the output at the end of the fourth round. We say that Π is a 1-rewinding sender secure protocol with delayed function selection for NC^1 circuits if it satisfies:*

- ***Delayed Function Selection.*** *The first and second message functions Π_1, Π_2 take as input the size of the function $f \in \mathsf{NC}^1$ to be securely computed and are otherwise, independent of the function description. The third round message from $\mathcal{R}$ contains the explicit description of the function f to be computed.*
- ***Receiver Security.*** *For every malicious PPT adversary $\mathcal{A}$ that corrupts the sender, there exists an expected polynomial (black-box) simulator $\mathsf{Sim}_{\mathcal{R}} = (\mathsf{Sim}^1_{\mathcal{R}}, \mathsf{Sim}^2_{\mathcal{R}})$ such that for all choices of honest receiver input $x_{\mathcal{R}}$ and the function $f \in \mathsf{NC}^1$, the joint distribution of the view of $\mathcal{A}$ and $\mathcal{R}$'s output in the real execution is computationally indistinguishable to the output of the ideal experiment described in Figure 3.*

[7]This is described in detail in Appendix A.

- ***1-Rewinding Sender Security.*** *For every malicious adversary $\mathcal{A}$ corrupting the receiver, there exists an expected polynomial time simulators $\mathsf{Sim}_{\mathcal{S}} = (\mathsf{Sim}_{\mathcal{S}}^1, \mathsf{Sim}_{\mathcal{S}}^2)$ such that for every choice of sender's input $x_{\mathcal{S}}$, we have:*
$$\mathsf{Expt}_1(\mathcal{A}, \Pi, x_{\mathcal{R}}, x_{\mathcal{S}}) \approx_c \mathsf{Expt}_2(\mathcal{A}, \mathsf{Sim}_{\mathcal{S}}, x_{\mathcal{R}}, x_{\mathcal{S}})$$
where Expt_1 *and* Expt_2 *are defined in Figure 4.*

- The honest receiver $\mathcal{R}$ sends $x_{\mathcal{R}}$ and f to the ideal functionality.
- Initialize $\mathcal{A}$ with uniform random tape r.
- $\mathsf{Sim}_{\mathcal{R}}^1$ on input f, interacts with $\mathcal{A}$ and outputs $\pi_1, \pi_2, x_{\mathcal{S}}$ and sk.
- Send $x_{\mathcal{S}}$ to the ideal functionality.
- $\mathsf{Sim}_{\mathcal{R}}^2$ on input sk, interacts with $\mathcal{A}$ and outputs π_3 and π_4. $\mathsf{Sim}_{\mathcal{R}}^2$ may send an abort to the ideal functionality.
- Output $(r, \pi_1, \pi_2, \pi_3, \pi_4)$ and the output of the honest $\mathcal{R}$.

Figure 3: Ideal Experiment in the Receiver Security Game

5 The (Multiparty) Simultaneous OT Protocol

In this section, we formally construct and prove security of round optimal watchlists that make black-box use of any public key encryption scheme with pseudorandom public keys. We prove the following theorem, that establishes the existence of simultaneous OT satisfying Definition 4.1.

Theorem 5.1. *Let λ denote the security parameter, and $m = m(\lambda), k = k(\lambda)$ be arbitrary polynomials. Then there exists a 4 round n-party (m, k) simultaneous OT protocol satisfying Definition 4.1 against static, malicious adversaries satisfying security with selective abort, that makes black-box use of any public key encryption with pseudorandom public keys.*

We prove Theorem 5.1 by first constructing non-malleable OT (Definition 5.3) in Section 5.1, and proving that this implies (multiparty) simultaneous OT (Definition 4.1) in Section 5.2.

Corollary given below follows immediately from Theorem 5.1 and [IKO$^+$11].

$\mathsf{Expt}_1(\mathcal{A}, \Pi, x_{\mathcal{S}}) = 1]$

- Initialize $\mathcal{A}$ with a uniform random tape s.
- $\pi_1 \leftarrow \mathcal{A}(1^\lambda; s)$.
- Choose $r \leftarrow \{0,1\}^\lambda$ uniformly at random and compute $\pi_2 \leftarrow \Pi_2(x_{\mathcal{S}}, \pi_1; r)$.
- $(f_0, \pi_3[0]), (f_1, \pi_3[1]) \leftarrow \mathcal{A}(\pi_2; s)$.
- $\pi_4[b] \leftarrow \Pi_4(x_{\mathcal{S}}, \pi_1, (f_b, \pi_3[b]); r)$ for $b \in \{0,1\}$.
- Output $(s, \pi_1, \pi_2, \{f_b, \pi_3[b], \pi_4[b]\}_{b\in\{0,1\}})$.

$\mathsf{Expt}_2(\mathcal{A}, \mathsf{Sim}_{\mathcal{S}}, x_{\mathcal{S}})$

- Initialize $\mathcal{A}$ with a uniform random tape s.
- $\mathsf{Sim}_{\mathcal{S}}^1$ interacts with $\mathcal{A}$ and produces (π_1, sk).
- $\mathsf{Sim}_{\mathcal{S}}^2$ on input sk interacts with $\mathcal{A}$ and produces a query $(x_{\mathcal{R}}, f_0, f_1)$ to be sent to the ideal functionality.
- On receiving $z_b = f_b(x_{\mathcal{R}}, x_{\mathcal{S}})$ from the ideal functionality, $\mathsf{Sim}_{\mathcal{S}}^2$ interacts with $\mathcal{A}$ and produces $(\pi_2, \{f_b, \pi_3[b], \pi_4[b]\}_{b\in\{0,1\}})$.
- Output $(s, \pi_1, \pi_2, \{f_b, \pi_3[b], \pi_4[b]\}_{b\in\{0,1\}})$.

Figure 4: Descriptions of Expt_1 and Expt_2.

Corollary 5.2. *There exists a four round n-party protocol satisfying Definition 4.1 that emulates arbitrary pairwise two-party functionalities with one-sided output, satisfying security with selevtive abort against static, malicious adversaries, and with black-box use of any public key encryption with pseudorandom public keys.*

5.1 ℓ Non-malleable m-choose-k OT: Construction and Analysis

In this section, we construct an ℓ non-malleable $\binom{m}{k}$ OT satisfying Definition 5.3. Our construction is described in Figure 5, and makes use of the following ingredients:

- A 4 round two-party secure computation protocol Π satisfying Definition 4.3 with delayed-function selection for NC^1 circuits and 1-rewinding sender security.
- An information-theoretic $m(\lambda) \cdot \ell(\lambda)$ non-malleable coding scheme satisfying Definition 3.1.
- A low-depth proof for P according to Definition 3.5.
- An existentially unforgeable signature scheme with algorithms denoted by $\mathsf{Signature.Setup}$, $\mathsf{Signature.Sign}$ and $\mathsf{Signature.Verify}$.

We describe our protocol formally in Figure 5. The correctness of this protocol follows from correctness of the underlying oblivious transfer, non-malleable codes and signature scheme. In what follows, we formally prove security.

First, we define *non-malleable* OT which secures against a man-in-the-middle adversary that possibly generates OT messages as a function of those generated by honest players. Loosely speaking, the non-malleability property ensures that no PPT adversarial sender can generate its OT inputs as a function of the private inputs of other (honest) senders.

Definition 5.3 (ℓ-non-malleable $\binom{m}{k}$ Oblivious Transfer). *An ℓ-non-malleable $\binom{m}{k}$ Oblivious Transfer protocol is a protocol between a sender $\mathcal{S}$ with inputs $\{\mathsf{x}_i\}_{i\in[m]}$ and a receiver $\mathcal{R}$ with input $K \subset [m]$ where $|K| = k$, that satisfies the following properties:*

- **Correctness.** *For every $i \in [m], \mathsf{x}_i \in \{0,1\}^\lambda$ and $K \subset [m]$ such that $|K| = k$,*

$$\mathsf{Out}_{\mathcal{R}}\langle \mathcal{S}(\{\mathsf{x}_i\}_{i\in[m]}), \mathcal{R}(K)\rangle = \{m_i\}_{i\in K}$$

- **Receiver Security (under Parallel Composition with Fixed Roles).** *For every PPT sender $\mathcal{S}^*$ and every pair K, K' of k-sized subsets of $[m]$, we require*

$$\mathsf{View}_{\mathcal{S}^*}\langle \mathcal{S}^*, \mathcal{R}(K)\rangle \approx_c \mathsf{View}_{\mathcal{S}^*}\langle \mathcal{S}^*, \mathcal{R}(K')\rangle$$

  Additionally, we require that there exists a PPT extractor $\mathsf{Sen.Ext}$ *that on input any transcript τ and with black-box access to any PPT sender $\mathcal{S}^*$ outputs $\{(\mathsf{x}^*_{i,j})\}_{i\in[m],j\in[\ell]}$ where $\mathsf{x}^*_{i,j}$ denotes the i^{th} implicit input used by $\mathcal{S}^*$ in the j^{th} session of τ (if any input is not well-defined, it outputs $\perp$ in its place)*[8].

[8]This property guarantees that it is possible to simulate the view of an adversarial sender that simultaneously participates in protocol sessions against multiple honest receivers. This property is satisfied by most natural rewinding-based protocols. This is different from non-malleability because here the adversary assumes the same role (i.e. sender) in all parallel protocol sessions, whereas in the case of non-malleability, the adversary is allowed to simultaneously assume the role of a sender in some sessions and a receiver in others.

- **Non-Malleability.** *Consider any PPT adversary (denoted by* MIM*) that interacts with upto* ℓ *senders* $\mathcal{S}_1, \ldots, \mathcal{S}_\ell$ *on the left, where for every* $j \in [\ell]$*,* $\mathcal{S}_j$ *has input* $\{\mathsf{x}_{i,j} \in \{0,1\}^n\}_{i\in[m]}$*, and upto* ℓ *receivers* $\mathcal{R}_1, \ldots, \mathcal{R}_\ell$ *on the right, where for every* $j \in [\ell]$*,* $\mathcal{R}_j$ *has input* K_j*.*

  We denote by $\mathsf{View}_{\mathsf{MIM}}\langle\{\mathcal{S}_j(\{\mathsf{x}_{i,j}\}_{i\in[m]})\}_{j\in[\ell]}, \{\mathcal{R}_j(K_j)\}_{j\in[\ell]}\rangle$ *the view of the* MIM *in this interaction, and denote the* i^{th} *implicit input used by the* MIM *in the* j^{th} *right session by* $\mathsf{x}'_{i,j}$ [9]*. We denote by* $\mathsf{Real}_{\mathsf{MIM}}\langle\{\mathcal{S}_j(\{\mathsf{x}_{i,j}\}_{i\in[m]})\}_{j\in[\ell]}, \{\mathcal{R}_j(K_j)\}_{j\in[\ell]}\rangle$ *the joint distribution of* $\{(\mathsf{x}'_{i,j})\}_{i\in[m],j\in[\ell]}$ *and* $\mathsf{View}_{\mathsf{MIM}}\langle\{\mathcal{S}_j(\{\mathsf{x}_{i,j}\}_{i\in[m]})\}_{j\in[\ell]}, \{\mathcal{R}_j(K_j)\}_{j\in[\ell]}\rangle$*. Then, we require that there exists a simulator-extractor pair,* $(\mathsf{Sim}_{\mathsf{OT}}, \mathsf{Ext}_{\mathsf{OT}})$ *that outputs*[10]

$$(\sigma_1, \overline{w}^{(1)}, \{\widetilde{K}_j\}_{j\in[\ell]}) \leftarrow \mathsf{Ext}_{\mathsf{OT}}^{\mathsf{MIM}}(\{K_j\}_{j\in[\ell]}),$$

$$\mathsf{Ideal}_{\mathsf{MIM}}(\{\mathsf{x}_{i,j}\}_{i\in[m],j\in[\ell]}, \{K_j\}_{j\in[\ell]}) \leftarrow \mathsf{Sim}_{\mathsf{OT}}^{\mathsf{MIM},\{\mathsf{OT}(\{\mathsf{x}_{i,j}\}_{i\in[m]},\cdot)\}_{j\in[\ell]}}(\sigma_1, \{\widetilde{K}_j\}_{j\in[\ell]}),$$

  where $\overline{w}^{(1)}$ *denotes the first round messages in a transcript generated by the* MIM *in its interaction with the extractor, and* σ *denotes the state of the extractor. We require that the adversary's "view" in the ideal world, which is a part of the distribution* $\mathsf{Ideal}_{\mathsf{MIM}}(\{\mathsf{x}_{i,j}\}_{i\in[m],j\in[\ell]}, \{K_j\}_{j\in[\ell]})$*, contain a transcript that has* $\overline{w}^{(1)}$ *as the first message, and that for all honest inputs* $\{\mathsf{x}_{i,j}\}_{i\in[m],j\in[\ell]}, \{K_j\}_{j\in[\ell]}$*,*

$$\mathsf{Real}_{\mathsf{MIM}}\langle\{\mathcal{S}_j(\{\mathsf{x}_{i,j}\}_{i\in[m]})\}_{j\in[\ell]}, \{\mathcal{R}_j(K_j)\}_{j\in[\ell]}\rangle \approx_c \mathsf{Ideal}_{\mathsf{MIM}}(\{\mathsf{x}_{i,j}\}_{i\in[m],j\in[\ell]}, \{K_j\}_{j\in[\ell]})$$

Next, we prove security according to Definition 5.3.

Theorem 5.4. *Let* λ *denote the security parameter, and* $m = m(\lambda), k = k(\lambda), \ell = \ell(\lambda)$ *be arbitrary polynomials. There exists a* 4 *round* ℓ *non-malleable* $\binom{m}{k}$ *oblivious transfer protocol satisfying Definition 5.3 that makes black-box use of any* 4 *round two-party secure computation protocol* Π *satisfying Definition 4.3, and any existentially unforgeable signature scheme.*

By relying on Theorem C.1, that shows how to build a 4 round two-party secure computation protocol Π satisfying Definition 4.3 based on black-box use of any public-key encryption with pseudo-random public keys, we obtain the following Corollary.

Corollary 5.5. *Let* λ *denote the security parameter, and* $m = m(\lambda), k = k(\lambda), \ell = \ell(\lambda)$ *be arbitrary polynomials. There exists a* 4 *round* ℓ *non-malleable* $\binom{m}{k}$ *OT protocol satisfying Definition 5.3 that makes black-box use of any public-key encryption with pseudo-random public keys.*

Proof of Theorem 5.4. We now consider a man-in-the-middle adversary that participates as an OT receiver in upto $\ell(\lambda)$ executions of this protocol on the right, and participates as an OT sender in upto $\ell(\lambda)$ executions on the left.

We will prove that there exists a PPT algorithm $\mathsf{Sim\text{-}Ext}$, that with black-box access to the MIM, to ℓ copies of the ideal OT functionality $\mathbf{OT} = \{\mathsf{OT}_j(\{\mathsf{x}_{i,j}\}_{i\in[m]}, \cdot)\}_{j\in[\ell]}$ and with input $\{K_j\}_{j\in[\ell]}$, simulates an execution of the protocol with the MIM and extracts all the inputs $\{(\{\widetilde{\mathsf{x}}_{i,j}\}_{i\in[m]})\}_{j\in[\ell]}$ used by the MIM in the executions where the MIM is sender. We will prove that $\mathsf{Sim\text{-}Ext}$ outputs $\mathsf{Ideal}_{\mathsf{MIM}}(\{\mathsf{x}_{i,j}\}_{i\in[m],j\in[\ell]}, \{K_j\}_{j\in[\ell]})$ according to Definition 5.3, such that

$$\mathsf{Real}_{\mathsf{MIM}}\langle\{\mathcal{S}_j(\{\mathsf{x}_{i,j}\}_{i\in[m]})\}_{j\in[\ell]}, \{\mathcal{R}_j(K_j)\}_{j\in[\ell]}\rangle \approx_c \mathsf{Ideal}_{\mathsf{MIM}}(\{\mathsf{x}_{i,j}\}_{i\in[m],j\in[\ell]}, \{K_j\}_{j\in[\ell]})$$

[9]If any of these is not well-defined, we denote it by $\perp$.

[10]We remark that we split the simulator-extractor into a pair of machines primarily for conceptual simplicity.

Inputs: Sender $\mathcal{S}$ has inputs $\{\mathsf{x}_j\}_{j\in m}$ and receiver $\mathcal{R}$ has input a set $K \subseteq [m]$ where $|K| = k$.

Protocol: $\mathcal{S}$ and $\mathcal{R}$ do the following.

1. $\mathcal{S}$ samples $(vk, sk) \leftarrow \mathsf{Signature.Setup}(1^\lambda)$, then does the following.
   - For each $i \in [\lambda], j \in [m]$, pick uniform randomness $r_{i,j}$ and compute $(\mathsf{L}_{i,j}, \mathsf{R}_{i,j}) = \mathsf{NM.Code}((vk|\mathsf{x}_j); r_{i,j})$.
   - Set $x = (vk, \{(\mathsf{L}_{i,j}, \mathsf{R}_{i,j}, \mathsf{x}_j)\}_{i\in[\lambda],j\in[m]})$ and $\mathcal{L} = \big\{(vk, \{(\mathsf{L}_{i,j}, \mathsf{R}_{i,j}, \mathsf{x}_j)\}_{i\in[\lambda],j\in[m]}) : \forall i \in [\lambda], j \in [m], \mathsf{NM.Decode}(\mathsf{L}_{i,j}, \mathsf{R}_{i,j}) = (vk|\mathsf{x}_j)\big\}$. Compute $\mathsf{ldp} = \mathsf{LDP.Prove}(x, \mathcal{L})$.
2. Both parties engage in the protocol Π to compute functionality $\mathcal{F}$ where:
   - $\mathcal{R}$ plays the receiver with input K committed in round 1 and delayed function $(\mathsf{c}_1, \ldots, \mathsf{c}_\lambda)$ that is chosen in round 3 by sampling each $\mathsf{c}_i \leftarrow \{0, 1\}$.
   - $\mathcal{S}$ plays the sender with input (x, ldp), where x is parsed as $(vk, \{\mathsf{x}_j, (\mathsf{L}_{i,j}, \mathsf{R}_{i,j})\}_{i\in[\lambda],j\in[m]}$.
   - The functionality $\mathcal{F}$ on input $(vk, \{\mathsf{x}_j, \mathsf{L}_{i,j}, \mathsf{R}_{i,j}\}_{i\in[\lambda],j\in[m]}, \mathsf{ldp}, K, \{\mathsf{c}_i\}_{i\in[\lambda]})$ generates an output as follows:
     - If $\mathsf{LDP.Verify}(x, \mathsf{ldp}) \neq 1$, output $\perp$.
     - Otherwise set $\mathsf{out} = vk, \{\mathsf{x}_j\}_{j\in K}$. Additionally, for every $i \in [\lambda]$, if $c_i = 0$, append $(\{\mathsf{L}_{i,j}\}_{j\in[m]})$ to out, else append $(\{\mathsf{R}_{i,j}\}_{j\in[m]})$ to out.
     - Output out.

   Additionally, $\mathcal{S}$ signs messages generated according to Π, denoted by (Π_2, Π_4). It sets $\sigma_2 = \mathsf{Signature.Sign}(\Pi_2, sk)$, $\sigma_4 = \mathsf{Signature.Sign}(\Pi_4, sk)$and sends (σ_2, σ_4) to $\mathcal{R}$.
3. $\mathcal{R}$ obtains output out and parses $\mathsf{out} = (vk, \{\mathsf{x}_j\}_{j\in K}, \cdot)$. It outputs $\{\mathsf{x}_j\}_{j\in K}$ iff $\mathsf{Signature.Verify}(\sigma_2, \Pi_2, vk) \wedge \mathsf{Signature.Verify}(\sigma_4, \Pi_4, vk) = 1$, otherwise outputs $\perp$.

Figure 5: $\ell(\lambda)$ Non-Malleable $m(\lambda)$-choose-$k(\lambda)$ Oblivious Transfer

To prove indistinguishability, we define a sequence of hybrid experiments, where the first one outputs the distribution $\mathsf{Real}_{\mathsf{MIM}}\langle\{\mathcal{S}_j(\{\mathsf{x}_{i,j}\}_{i\in[m]})\}_{j\in[\ell]}, \{\mathcal{R}_j(K_j)\}_{j\in[\ell]}$ and the final one outputs the distribution $\mathsf{Ideal}_{\mathsf{MIM}}(\{\mathsf{x}_{i,j}\}_{i\in[m],j\in[\ell]}, \{K_j\}_{j\in[\ell]})$. The simulation strategy is identical to that of the challenger in the final hybrid.

Hyb_0 : This corresponds to an execution of the MIM with ℓ honest senders $\{\mathcal{S}_j\}_{j\in[\ell]}$ on the left, each using inputs $\{\mathsf{x}_{i,j}\}_{i\in[m]}$ respectively and ℓ honest receivers on the right with inputs $(\{K_j\}_{j\in[\ell]})$ respectively. The output of this hybrid is $\mathsf{Real}_{\mathsf{MIM}}\langle\{\mathcal{S}_j(\{\mathsf{x}_{i,j}\}_{i\in[m]})\}_{j\in[\ell]}, \{\mathcal{R}_j(K_j)\}_{j\in[\ell]}\rangle$.

Hyb_1 : This experiment modifies Hyb_1 by introducing an additional abort condition, and changing the way the adversary's inputs $\{\widetilde{\mathsf{x}}_i^j\}_{j\in[\ell],i\in[m]}$ are computed.

Specifically, the experiment first executes the complete protocol corresponding to the real execution of the MIM exactly as in Hyb_0 to obtain $\mathsf{View}_{\mathsf{MIM}}\langle\{\mathcal{S}_j(\{\mathsf{x}_{i,j}\}_{i\in[m]})\}_{j\in[\ell]}, \{\mathcal{R}_j(K_j)\}_{j\in[\ell]}\rangle$.

Let $p(\lambda)$ denote the probability that the MIM completes this execution without aborting. Set

$\gamma(\lambda) = \max\left(\lambda, p^{-2}(\lambda)\right)$. With the first two rounds of the transcript fixed, the rewind the right execution up to $\gamma(\lambda)$ times, picking inputs $(\mathsf{c}_1^j, \ldots, \mathsf{c}_\lambda^j)$ for each of the ℓ receivers $\{\mathcal{R}_j\}_{j\in[\ell]}$ independently and uniformly at random in every run. If there exists a rewinding thread where the MIM completes the protocol execution, denote the inputs chosen by the challenger on behalf of the honest receiver in this rewinding thread by $(\mathsf{c}'^j_1, \ldots, \mathsf{c}'^j_\lambda)$. For every $j \in [\ell]$, let index $\alpha_j \in [\lambda]$ be such that $\mathsf{c}^j_{\alpha_j} = 0$ and $\mathsf{c}'^j_{\alpha_j} = 1$. Additionally for every $j \in [\ell], i \in [m]$, use $(\widetilde{\mathsf{L}}^j_{\alpha_j,i}, \widetilde{\mathsf{R}}^j_{\alpha_j,i})$ obtained as output from the main and rewinding executions respectively to compute $\widetilde{\mathsf{x}}^j_i = \mathsf{NM.Decode}(\widetilde{\mathsf{L}}^j_{\alpha_j,i}, \widetilde{\mathsf{R}}^j_{\alpha_j,i})$.

If no such rewinding thread exists, or if there exists $j \in [\ell]$ for which there does not exist $\alpha \in [\lambda]$ such that $\mathsf{c}^j_\alpha = 0$ and $\mathsf{c}'^j_\alpha = 1$, then set $\widetilde{\mathsf{x}}^j_i = \perp$ for all $i \in [m]$. Now, the output of this hybrid is the joint distribution

$$\mathsf{View}_{\mathsf{MIM}}\langle\{\mathcal{S}_j(\{\mathsf{x}^j_i\}_{i\in[m]})\}_{j\in[\ell]}, \{\mathcal{R}^j(K^j)\}_{j\in[\ell]}\rangle, \{\widetilde{\mathsf{x}}^j_i\}_{j\in[\ell],i\in[m]}.$$

Lemma 5.6. *For every unbounded distinguisher $\mathcal{D}$ and large enough $\lambda \in \mathbb{N}$,*

$$\left|\Pr[\mathcal{D}(\mathsf{Hyb}_0) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}_1) = 1]\right| = \mathsf{negl}(\lambda)$$

Proof. Since the MIM's inputs $\{(\widetilde{\mathsf{x}}^j_i\}_{j\in[\ell]}$ are committed in round 2 of the protocol, then conditioned on the adversary providing a non-aborting transcript in a rewinding execution in Hyb_1, by simulation security of the 2pc, $\{(\widetilde{\mathsf{x}}^j_i\}_{j\in[\ell]}$ are correctly extracted.

Therefore, to prove this lemma it suffices to show that such a rewinding execution (with a non-aborting transcript) can be found within $\gamma(\lambda)$ attempts, except with probability $\mathsf{negl}(\lambda)$. To see this, we observe that the probability of a non-aborting transcript is $p(\lambda)$, and therefore, the probability that all $\gamma(\lambda)$ trials abort is $(1-p(\lambda))^{\gamma(\lambda)} = \mathsf{negl}(\lambda)$. □

Hyb_2: This experiment modifies Hyb_1 to execute the simulator of Π in all sessions where the MIM is a receiver.

In more detail, in all executions j where MIM is a receiver, instead of the honest sender strategy with input $\{\mathsf{x}^j_i\}_{i\in[m],j\in[\ell]}$, this hybrid executes the simulator $\mathsf{Sim\text{-}2PC}_{\mathsf{Sen}}^{\mathsf{MIM},\mathcal{F}(\mathsf{inp}_{\mathcal{S}^j},\cdot)}$ where

$$\mathsf{inp}_{\mathcal{S}^j} = (\{\mathsf{x}^j_i, \mathsf{L}^j_{1,i}, \ldots, \mathsf{L}^j_{\lambda,i}, \mathsf{R}^j_{1,i}, \ldots, \mathsf{R}^j_{\lambda,i}\}_{i\in[m]}).$$

$\mathsf{Sim\text{-}2PC}_{\mathsf{Sen}}$ expects round 1 and round 3 messages from the MIM, and the MIM in turn expects corresponding messages from the receiver in the right execution. Receiver messages for the right execution are generated using honest receiver strategy with inputs K^j fixed, and inputs $c^j_1, \ldots, c^j_\lambda$ chosen uniformly at random, exactly as in Hyb_1. Denote the view of the MIM by

$$\mathsf{View}_{\mathsf{Sim}^{\{\mathcal{F}(\mathsf{inp}_{\mathcal{S}^j},\cdot)\}_{j\in[\ell]}}}\langle\{\mathcal{R}^j(K^j)\}_{j\in[\ell]}\rangle,$$

where for every $j \in [\ell]$, $\mathsf{inp}_{\mathcal{S}^j}$ is as defined above.

Next, with the first two rounds of the transcript fixed, the challenger rewinds the right execution up to $\ell(\lambda)$ times, picking inputs $(\mathsf{c}^j_1, \ldots, \mathsf{c}^j_\lambda)$ independently and uniformly at random in every run, and generating messages in the left execution by running the simulator $\mathsf{Sim\text{-}2PC}_{\mathsf{Sen}}$ on behalf of $\mathcal{R}^j$.

If there exists a rewinding execution where the MIM completes the protocol, denote the inputs chosen by the challenger on behalf of the honest receiver in this rewinding thread by $(\mathsf{c}'^j_1, \ldots, \mathsf{c}'^j_\lambda)$. For every $j \in [\ell]$, let index $\alpha_j \in [\lambda]$ be such that $\mathsf{c}^j_{\alpha_j} = 0$ and $\mathsf{c}'^j_{\alpha_j} = 1$. Additionally for every

$j \in [\ell], i \in [m]$, use $(\widetilde{\mathsf{L}}^j_{\alpha_j,i}, \widetilde{\mathsf{R}}^j_{\alpha_j,i})$ obtained as output from the main and rewinding executions respectively to compute $\widetilde{\mathsf{x}}^j_i = \mathsf{NM.Decode}(\widetilde{\mathsf{L}}^j_{\alpha_j,i}, \widetilde{\mathsf{R}}^j_{\alpha_j,i})$. If no such rewinding thread exists, or if there exists $j \in [\ell]$ for which there does not exist $\alpha \in [\lambda]$ such that $\mathsf{c}^j_\alpha = 0$ and $\mathsf{c'}^j_\alpha = 1$, then set $\widetilde{\mathsf{x}}^j_i = \perp$ for all $i \in [m]$. The output of this hybrid is the joint distribution:

$$\mathsf{View}_{\mathsf{Sim}^{\{\mathcal{F}(\mathsf{inp}_{\mathcal{S}^j},\cdot)\}_{j\in[\ell]}}} \langle \{\mathcal{R}^j(K^j)\}_{j\in[\ell]} \rangle, \{\widetilde{\mathsf{x}}^j_i\}_{j\in[\ell], i\in[m]},$$

where for every $j \in [\ell]$, $\mathsf{inp}_{\mathcal{S}^j}$ is as defined above.

Lemma 5.7. *Assuming* 1*-rewinding secure two party computation according to Definition 4.3, for every PPT distinguisher* $\mathcal{D}$ *and large enough* $\lambda \in \mathbb{N}$,

$$\Big| \Pr[\mathcal{D}(\mathsf{Hyb}_1) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}_2) = 1] \Big| = \mathsf{negl}(\lambda)$$

Proof. To prove this lemma, we consider a sequence of sub-hybrids $\mathsf{Hyb}_{1,0}, \mathsf{Hyb}_{1,1}, \ldots \mathsf{Hyb}_{1,\ell}$ where for every $j \in [\ell]$, $\mathsf{Hyb}_{1,j}$ is identical to $\mathsf{Hyb}_{1,j-1}$, except that instead of executing the honest sender strategy using honest sender inputs $\{m^j_i\}_{i\in[m]}$, we execute the simulator in the j^{th} left execution, where $\mathsf{Sim\text{-}2PC}^{\mathsf{MIM},\mathcal{F}(\mathsf{inp}_{\mathcal{S}^j},\cdot)}_{\mathsf{Sen}}$ where

$$\mathsf{inp}_{\mathcal{S}^j} = (\{\mathsf{x}^j_i, \mathsf{L}^j_{1,i}, \ldots, \mathsf{L}^j_{\lambda,i}, \mathsf{R}^j_{1,i}, \ldots, \mathsf{R}^j_{\lambda,i}\}_{i\in[m]})$$

Suppose the lemma is not true. Then for every large enough $\lambda \in \mathbb{N}$ there exists $j^*(\lambda) \in [\ell(\lambda)]$, a polynomial $p(\cdot)$ and a distinguisher $\mathcal{D}$ such that for infinitely many $\lambda \in \mathbb{N}$,

$$\Big| \Pr[\mathcal{D}(\mathsf{Hyb}_{1,j^*-1}) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}_{1,j^*}) = 1] \Big| \geq \frac{1}{q(\lambda)}$$

We derive a contradiction by building a reduction $\mathcal{A}$ that on input λ, obtains $j^*(\lambda)$ as advice and with black-box access to the MIM and to $\mathcal{D}$ contradicts 1-rewinding security of the two party computation protocol. $\mathcal{A}$ proceeds as follows:

- $\mathcal{A}$ first creates receiver $\mathcal{R}'$ that interacts with the external challenger as follows.
  - Generate the first round messages according to receiver strategy with inputs $\{K^j\}_{j\in[\ell]}$ for the right execution. Obtain first round messages from the MIM, and output the MIM's message in the j^{*th} left execution to the challenger of the 2pc.
  - Obtain the second round message for the left execution externally from the 2pc challenger, and forward this to the MIM as $\mathcal{S}^{j^*}$'s message in the j^{*th} left execution. Obtain the second round message from the MIM for the right execution.
  - Generate the third round message for the right execution according to honest receiver strategy, and obtain the third round message from the MIM. Output the MIM's message in left session j^* to the challenger.
  - Obtain the fourth round message for the left execution externally from the challenger, and forward this to the MIM as $\mathcal{S}$'s message in the j^{*th} left execution. Obtain the fourth round message from the MIM for the right execution.
- Next, $\mathcal{A}$ rewinds $\mathcal{R}'$ once, as follows.
  - Generate the third round message according to honest receiver strategy, and obtain the third round message from the MIM. Output the MIM's message in session j^* to the challenger.

  - Obtain the fourth round message for the left execution externally from the challenger, and forward this to the MIM as $\mathcal{S}$'s message in the j^{*th} left execution. Obtain the fourth round message from the MIM from the left execution.
  - If none of the executions abort, for every $j \in [\ell]$, find $\alpha_j \in [\lambda]$ such that $\mathsf{c}^j_{\alpha_j} = 0$ and $\mathsf{c'}^j_{\alpha_j} = 1$. and use it to compute $\widetilde{\mathsf{x}}^j_i = \mathsf{NM.Decode}(\widetilde{\mathsf{L}}^j_{\alpha_j,i}, \widetilde{\mathsf{R}}^j_{\alpha_j,i})$ for $i \in [m], j \in [\ell]$. Else, set $\widetilde{\mathsf{x}}^j_i = \bot$ for $i \in [m], j \in [\ell]$

- $\mathcal{A}$ outputs the entire view of $\mathcal{R}'$ together with $\{\widetilde{\mathsf{x}}^j_i\}_{i\in[m],j\in[\ell]}$. If the challenger used honest sender messages, we denote the distribution output by $\mathcal{A}$ in this experiment by Dist_1 and if the challenger used simulated messages, we denote the distribution output by $\mathcal{A}$ in this experiment by Dist_2.

If the challenger's messages correspond to the real sender $\mathcal{S}$, then the distribution output by $\mathcal{A}$ conditioned on not aborting corresponds to Hyb_{1,j^*-1}, and if the challenger's messages correspond to $\mathsf{Sim\text{-}2PC}_{\mathsf{Sen}}$, then the distribution output by $\mathcal{A}$ conditioned on not aborting corresponds to Hyb_{1,j^*}.

By assumption, for infinitely many $\lambda \in \mathbb{N}$,

$$\Big|\Pr[\mathcal{D}(\mathsf{Hyb}_1) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}_2) = 1]\Big| = \frac{1}{q(\lambda)}$$

Since the MIM completes any run of the protocol without aborting with probability at least $p(\lambda)$, and because aborts are independent of the distinguishing advantage, for infinitely many $\lambda \in \mathbb{N}$:

$$\Big|\Pr[\mathcal{D} = 1 \ \wedge \ \neg\mathsf{abort}|\mathsf{Hyb}_1] - \Pr[\mathcal{D} = 1 \ \wedge \ \neg\mathsf{abort}|\mathsf{Hyb}_2]\Big| \geq \frac{1}{p(\lambda)\cdot q(\lambda)}$$

where $\neg\mathsf{abort}$ denotes the event that an execution that is completed in the main thread, is also completed without aborting in one rewinding execution.

This implies that for infinitely many $\lambda \in \mathbb{N}$:

$$\Big|\Pr[\mathcal{D}(\mathsf{Dist}_1) = 1] - \Pr[\mathcal{D}(\mathsf{Dist}_2) = 1]\Big| \geq \frac{1}{p(\lambda)\cdot q(\lambda)},$$

which implies that $\mathcal{D}$ contradicts 1-rewinding security of the two party computation protocol. □

Hyb_3: This hybrid is the same as Hyb_2 except whenever the challenger obtains as output a verification key in one of the right sessions that is identical to a verification key used in one of the left sessions, the hybrid outputs $\bot$. By existential unforgeability of the signature scheme, given any PPT adversary MIM, Hyb_2 and Hyb_3 are statistically indistinguishable.

Hyb_4: This hybrid is the same as Hyb_3 except that $\mathsf{inp}_{\mathcal{S}^j}$ is set differently. Specifically, for every $j \in [\ell], i \in [m]$ and $\alpha \in [\lambda]$, we set $(\mathsf{L}^j_{\alpha,i}, \mathsf{R}^j_{\alpha,i}) \leftarrow \mathsf{NM.Sim}(1^{p(\lambda)})$, and set

$$\mathsf{inp}_{\mathcal{S}^j} = (\{\mathsf{x}^j_i, \mathsf{L}^j_{1,i}, \ldots, \mathsf{L}^j_{\lambda,i}, \mathsf{R}^j_{1,i}, \ldots, \mathsf{R}^j_{\lambda,i}\}_{i\in[m]}).$$

We note that at this point, the functionality $\{\mathcal{F}(\mathsf{inp}_{\mathcal{S}^j}, \cdot)\}_{j\in[\ell]}$ can be perfectly simulated with access to the ideal functionality $\{\mathsf{OT}^j(\mathsf{x}^j_i, \mathsf{x}^j_i, \cdot)\}_{j\in[\ell]}$. Therefore, the output of this hybrid is identical to the ideal view $\mathsf{Ideal}_{\mathsf{MIM}}(\{\mathsf{x}^j_i\}_{i\in[m],j\in[\ell]}, \{K^j\}_{j\in[\ell]})$.

Lemma 5.8. *Assuming $m(\lambda) \cdot \ell(\lambda)$ symmetric non-malleable codes satisfying Definition 3.1, for every unbounded distinguisher $\mathcal{D}$ and large enough $\lambda \in \mathbb{N}$,*

$$\Big|\Pr[\mathcal{D}(\mathsf{Hyb}_4) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}_3) = 1]\Big| = \mathsf{negl}(\lambda)$$

Proof. We prove indistinguishability between Hyb_3 and Hyb_4 by considering a sequence of sub-hybrids, $\{\mathsf{Hyb}_{3,i,j,k}\}_{i\in[1,m],j\in[1,\ell],k\in[0,\lambda]}$ where:

- $\mathsf{Hyb}_3 = \mathsf{Hyb}_{3,0,\ell,\lambda}$, $\mathsf{Hyb}_4 = \mathsf{Hyb}_{3,m,\ell,\lambda}$,
- for $i \in [m]$, $\mathsf{Hyb}_{3,i-1,\ell,\lambda} = \mathsf{Hyb}_{3,i,1,0}$
- for $j \in [\ell]$, $\mathsf{Hyb}_{3,i,j-1,\lambda} = \mathsf{Hyb}_{3,i,j,0}$,
- for every $i \in [m], j \in [\ell], k \in [\lambda]$, $\mathsf{Hyb}_{3,i,j,k}$ is identical to $\mathsf{Hyb}_{3,i,j,k-1}$ except that $\mathsf{Hyb}_{3,i,j,k}$ samples $(\mathsf{L}^j_{k,i}, \mathsf{R}^j_{k,i}) \leftarrow \mathsf{NM.Code}(0)$.

Suppose the lemma is not true. Then there exists $i^* \in [m], j^* \in [\ell], k^* \in [\lambda]$, an unbounded distinguisher $\mathcal{D}$ and a polynomial $p(\cdot)$ such that for large enough $\lambda \in \mathbb{N}$,

$$\Big|\Pr[\mathcal{D}(\mathsf{Hyb}_{3,i^*,j^*,k^*}) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}_{3,i^*,j^*,k^*-1}) = 1]\Big| = \frac{1}{p(\lambda)} \tag{1}$$

We now define a pair of tampering functions $(f_{\mathsf{MIM}}, g_{\mathsf{MIM}})$, and additional function h_{MIM} as follows:

- $f_{\mathsf{MIM}}, g_{\mathsf{MIM}}$ and h_{MIM} share common state that is generated as follows:
  - Execute $\mathsf{Sim\text{-}2PC}^{\mathsf{MIM}}_{\mathsf{Sen}}$, using honest $\mathcal{R}$ strategy in the right executions with input $\{K^j\}_{j\in[\ell]}$ and uniformly chosen $\{c^j_1, \ldots c^j_\lambda\}_{j\in[\ell]}$, until $\mathsf{Sim\text{-}2PC}_{\mathsf{Sen}}$ generates a query to the ideal functionality $\mathcal{F}$ at the end of round 3.
  - At this point, $\mathsf{Sim\text{-}2PC}^{\mathsf{MIM}}_{\mathsf{Sen}}$ outputs a view and transcript of the MIM until the third round, as well as $\{\widetilde{K}^j, \widetilde{\mathsf{c}}^j_1, \ldots, \widetilde{\mathsf{c}}^j_\lambda\}_{j\in[\ell]}$ that correspond to the receiver's inputs in the left execution.
  - Rewind the third round once with uniformly and independently chosen $\{\mathsf{c}'^j_1, \ldots, \mathsf{c}'^j_\lambda\}_{j\in[\ell]}$. If for every $j \in [\ell(\lambda)]$, there exists $\alpha_j \in [\lambda]$ such that $\mathsf{c}^j_{\alpha_j} = 0$ and $\mathsf{c}'^j_{\alpha_j} = 1$, continue, otherwise abort.
  - Obtain the rewinding view (with the same prefix of the first two rounds), as well as $(\overline{\mathsf{c}}_1, \ldots, \overline{\mathsf{c}}_n)$ that correspond to the receiver's input in the left session in this rewinding execution. If $\widetilde{\mathsf{c}}^j_k \neq \overline{\mathsf{c}}^j_k$, continue. Otherwise, abort.
  - Generate $(\mathsf{L}^j_{k,i}, \mathsf{R}^j_{k,i})$ for every $(i,j,k) \in [m]\times[\ell]\times[\lambda]\backslash\{i^*, j^*, k^*\}$ according to $\mathsf{Hyb}_{3,i^*,j^*,k^*-1}$ (this is identical to setting them according to $\mathsf{Hyb}_{3,i^*,j^*,k^*}$).
  - Output the view of the MIM until round 3 in the main the rewinding threads, including (i^*, j^*, k^*), the values $(\mathsf{L}^j_{k,i}, \mathsf{R}^j_{k,i})_{(i,j,k)\in[m]\times[\ell]\times[\lambda]\backslash\{i^*,j^*,k^*\}}$.
  - Additionally, output the receiver's inputs $\{\widetilde{K}^j, \widetilde{\mathsf{c}}^j_1, \ldots, \widetilde{\mathsf{c}}^j_\lambda\}_{j\in[\ell]}$ and the sender's inputs $\{sk^j, vk^j, \{\mathsf{x}^j_i\}_{i\in[m]}\}_{j\in[\ell]}$.

- Next, the function h_{MIM} on input L, sets $\mathsf{L}^{j^*}_{k^*,i^*} = \mathsf{L}, \mathsf{R}^{j^*}_{k^*,i^*} = 0$.

  Now, using hardwired values $\{vk^j, \{\mathsf{x}^j_i\}_{i\in[m]}\}_{j\in[\ell]}$, $\{\widetilde{K}^j, \widetilde{\mathsf{c}}^j_1, \ldots, \widetilde{\mathsf{c}}^j_\lambda\}_{j\in[\ell]}$ as well as the values $(\mathsf{L}^j_{k,i}, \mathsf{R}^j_{k,i})_{(i,j,k)\in[m]\times[\ell]\times[\lambda]\setminus\{i^*,j^*,k^*\}}$, it computes

  $$\mathsf{out} = \{\mathcal{F}(\{\mathsf{x}^j_i, \mathsf{L}^j_{k,i}, \mathsf{R}^j_{k,i}\}_{i\in[m],k\in[\lambda]}, \widetilde{K}^j, \{\widetilde{\mathsf{c}}^j_k\}_{k\in[\lambda]})\}_{j\in[\ell]}.$$

  It then invokes $\mathsf{Sim\text{-}2PC_{Sen}}$ on out to generate the fourth round message of the protocol transcript in the main thread if $\widetilde{\mathsf{c}}^{j^*}_{k^*} = 0$, and generates the fourth round message of the protocol transcript in the rewinding thread if $\bar{\mathsf{c}}^{j^*}_{k^*} = 0$. It outputs the resulting transcript as the view of the MIM.

- The function f_{MIM} on input L, sets $\mathsf{L}^{j^*}_{k^*,i^*} = \mathsf{L}, \mathsf{R}^{j^*}_{k^*,i^*} = 0$.

  Now, using hardwired values $\{vk^j, \{\mathsf{x}^j_i\}_{i\in[m]}\}_{j\in[\ell]}$, $\{\widetilde{K}^j, \widetilde{\mathsf{c}}^j_1, \ldots, \widetilde{\mathsf{c}}^j_\lambda\}_{j\in[\ell]}$ as well as the values $(\mathsf{L}^j_{k,i}, \mathsf{R}^j_{k,i})_{(i,j,k)\in[m]\times[\ell]\times[\lambda]\setminus\{i^*,j^*,k^*\}}$, it computes

  $$\mathsf{out} = \{\mathcal{F}(\{\mathsf{x}^j_i, \mathsf{L}^j_{k,i}, \mathsf{R}^j_{k,i}\}_{i\in[m],k\in[\lambda]}, \widetilde{K}^j, \{\widetilde{\mathsf{c}}^j_k\}_{k\in[\lambda]})\}_{j\in[\ell]}.$$

  It then invokes $\mathsf{Sim\text{-}2PC_{Sen}}$ on out to generate the fourth round message of the protocol transcript in the main thread if $\widetilde{\mathsf{c}}^{j^*}_{k^*} = 0$, and generates the fourth round message of the protocol transcript in the rewinding thread if $\bar{\mathsf{c}}^{j^*}_{k^*} = 0$. It outputs the values $\{\mathsf{L}^j_{\alpha_j,i}\}_{i\in[m],j\in[\ell]}$ or $\{\mathsf{R}^j_{\alpha_j,i}\}_{i\in[m],j\in[\ell]}$ obtained from the MIM.

- The function g_{MIM} on input R, sets $\mathsf{R}^{j^*}_{k^*,i^*} = \mathsf{R}, \mathsf{L}^{j^*}_{k^*,i^*} = 0$.

  Now, using hardwired values $\{vk^j, \{\mathsf{x}^j_i\}_{i\in[m]}\}_{j\in[\ell]}$, $\{\widetilde{K}^j, \widetilde{\mathsf{c}}^j_1, \ldots, \widetilde{\mathsf{c}}^j_\lambda\}_{j\in[\ell]}$ as well as the values $(\mathsf{L}^j_{k,i}, \mathsf{R}^j_{k,i})_{(i,j,k)\in[m]\times[\ell]\times[\lambda]\setminus\{i^*,j^*,k^*\}}$, it computes

  $$\mathsf{out} = \{\mathcal{F}(\{\mathsf{x}^j_i, \mathsf{L}^j_{k,i}, \mathsf{R}^j_{k,i}\}_{i\in[m],k\in[\lambda]}, \widetilde{K}^j, \{\widetilde{\mathsf{c}}^j_k\}_{k\in[\lambda]})\}_{j\in[\ell]}.$$

  It then invokes $\mathsf{Sim\text{-}2PC_{Sen}}$ on out to generate the fourth round message of the protocol transcript in the main thread if $\widetilde{\mathsf{c}}^{j^*}_{k^*} = 1$, and generates the fourth round message of the protocol transcript in the rewinding thread if $\bar{\mathsf{c}}^{j^*}_{k^*} = 1$. It outputs the values $\{\mathsf{L}^j_{\alpha_j,i}\}_{i\in[m],j\in[\ell]}$ or $\{\mathsf{R}^j_{\alpha_j,i}\}_{i\in[m],j\in[\ell]}$ obtained from the MIM.

By Definition 3.1 of 1-many augmented non-malleable codes,

$$\Big(\mathsf{L}, \mathsf{NM.Decode}\big(f_{\mathsf{MIM}}(\mathsf{L}), g_{\mathsf{MIM}}(\mathsf{R})\big)\Big|(\mathsf{L}, \mathsf{R} \leftarrow \mathsf{NM.Code}(\mathsf{x}^{j^*}_{i^*}))\Big) \approx_\epsilon$$
$$\Big(\mathsf{L}, \mathsf{NM.Decode}\big(f_{\mathsf{MIM}}(\mathsf{L}), g_{\mathsf{MIM}}(\mathsf{R})\big)\Big|(\mathsf{L}, \mathsf{R} \leftarrow \mathsf{NM.Code}(0))\Big) \text{ and}$$
$$\Big(\mathsf{L}, \mathsf{NM.Decode}\big(g_{\mathsf{MIM}}(\mathsf{R}), f_{\mathsf{MIM}}(\mathsf{L})\big)\Big|(\mathsf{L}, \mathsf{R} \leftarrow \mathsf{NM.Code}(\mathsf{x}^{j^*}_{i^*}))\Big) \approx_\epsilon$$
$$\Big(\mathsf{L}, \mathsf{NM.Decode}\big(g_{\mathsf{MIM}}(\mathsf{R}), f_{\mathsf{MIM}}(\mathsf{L})\big)\Big|(\mathsf{L}, \mathsf{R} \leftarrow \mathsf{NM.Code}(0))\Big)$$

By the data processing inequality, this implies that for every function $h(\cdot)$,

$$\Big(h(\mathsf{L}), \mathsf{NM.Decode}\big(f_{\mathsf{MIM}}(\mathsf{L}), g_{\mathsf{MIM}}(\mathsf{R})\big)\Big|(\mathsf{L},\mathsf{R} \leftarrow \mathsf{NM.Code}(\mathsf{x}_{i^*}^{j^*}))\Big) \approx_\epsilon$$
$$\Big(h(\mathsf{L}), \mathsf{NM.Decode}\big(f_{\mathsf{MIM}}(\mathsf{L}), g_{\mathsf{MIM}}(\mathsf{R})\big)\Big|(\mathsf{L},\mathsf{R} \leftarrow \mathsf{NM.Code}(0))\Big) \text{ and}$$
$$\Big(h(\mathsf{L}), \mathsf{NM.Decode}\big(g_{\mathsf{MIM}}(\mathsf{R}), f_{\mathsf{MIM}}(\mathsf{L})\big)\Big|(\mathsf{L},\mathsf{R} \leftarrow \mathsf{NM.Code}(\mathsf{x}_{i^*}^{j^*}))\Big) \approx_\epsilon$$
$$\Big(h(\mathsf{L}), \mathsf{NM.Decode}\big(g_{\mathsf{MIM}}(\mathsf{R}), f_{\mathsf{MIM}}(\mathsf{L})\big)\Big|(\mathsf{L},\mathsf{R} \leftarrow \mathsf{NM.Code}(0))\Big)$$

Setting $h = h_{\mathsf{MIM}}$, for f_{MIM} and g_{MIM} defined above, these distributions correspond to the outputs of $\mathsf{Hyb}_{3,i^*,j^*,k^*-1}$ and $\mathsf{Hyb}_{3,i^*,j^*,k^*}$ respectively, whenever $\widetilde{c}_{k^*}^{j^*} \neq \overline{c}_{k^*}^{j^*}$. Whenever $\widetilde{c}_{k^*}^{j^*} = \overline{c}_{k^*}^{j^*}$, the distributions $\mathsf{Hyb}_{3,i^*,j^*,k^*-1}$ and $\mathsf{Hyb}_{3,i^*,j^*,k^*}$ are $\epsilon(\lambda)$-statistically indistinguishable because they jointly only depend on one of the shares, L or R. Since $\epsilon(\lambda) = \mathsf{negl}(\lambda)$, this contradicts Equation (1), completing our proof. □

5.2 From Non-Malleable OT to (Multiparty) Simultaneous OT

Theorem 5.9. *Let $\ell = \ell(\lambda), k = k(\lambda), m = m(\lambda)$ and $n = n(\lambda)$ be arbitrary polynomials such that for large enough $\lambda \in N$, $\ell(\lambda) \geq n^2(\lambda)$. Then there exists a black-box construction of an n-Party (m,k) simultaneous OT protocol according to Definition 4.1 from any ℓ non-malleable $\binom{m}{k}$ oblivious transfer protocol that satisfies Definition 5.3.*

Proof. We begin by constructing an n-party (m,k) (multiparty) simultaneous OT protocol from any ℓ non-malleable $\binom{m}{k}$ oblivious transfer protocol.

Construction. First, we establish some basic notation.

- For $i \in [n]$, denote P_i's OT input by $\big(\{X_{i,j}\}_{j\in[n]\setminus\{i\}}, \{y_{i,j}\}_{j\in[n]\setminus\{i\}}\big)$ where:
  - For every $j \in [n] \setminus \{i\}$, $y_{i,j} \subset [m]$ such that $|y_{i,j}| = k$, and
  - For every $j \in [n] \setminus \{i\}$, $X_{i,j} = (s_{i,j,1}, \ldots, s_{i,j,m})$.
- Denote an instance of the ℓ non-malleable $\binom{m}{k}$ oblivious transfer protocol by Π_{OT}.

The construction itself is straightforward: for every $i,j \in [n] \times [n]$ where $i \neq j$, P_i and P_j execute an instance of Π_{OT} with P_i as sender with input $\{s_{i,j,k}\}_{k\in[m]}$ and P_j as receiver with input $y_{j,i}$.

Correctness. Correctness of the n-party $\binom{m}{k}$ (multiparty) simultaneous OT follows from the correctness of the $\binom{m}{k}$ OT protocol.

Proof of Security (Sketch). We describe how to prove a stronger security property of the resulting (multiparty) simultaneous OT protocol: that for every corrupted subset $M \subset [n]$, there exists a simulator $\mathsf{Sim}_{\mathsf{mpOT}}$ such that for every (malicious) non-uniform adversary $\mathcal{A}$ that corrupts $\{P_i\}_{i \in M}$, and for every choice of inputs $\left(\{X_{i,j}\}_{j\in[n]\setminus\{i\}}, \{y_{i,j}\}_{j\in[n]\setminus\{i\}}\right)$, the "real" and "ideal" views defined below, are indistinguishable.

$$\begin{aligned}&\Big(\mathsf{View}_{\mathcal{A}}\left(\mathsf{mpOT}(\{x_i\}_{i\in[n]\setminus M})\right), \{\mathsf{out}_j\left(\mathsf{mpOT}(\{x_i\}_{i\in[n]\setminus M})\right)\}_{j\in[n]\setminus M}\Big)\\ \approx_c \quad &\Big(\overline{w}^{(1-4)}, \{\mathsf{out}_j\left(\mathcal{F}_{\mathsf{OT}}^{n\cdot(n-1)}(\{x_i\}_{i\in[n]\setminus M}, \{\widetilde{x}_i\}_{i\in M})\right)\}_{j\in[n]\setminus M}\Big)\end{aligned}$$

where

$$\begin{aligned}&(\sigma_1, \overline{w}^{(1)}, \{\{\widetilde{K}_{i,j}\}_{j\in[n]\setminus\{i\}}\}_{i\in M}) \leftarrow (\mathsf{Sim}_{\mathsf{mpOT}})^{\mathcal{A}};\\ &\quad(\sigma_4, \overline{w}^{(2-4)}, \{\{\widetilde{s}_{i,j}\}_{j\in[n]\setminus\{i\}}\}_{i\in M}) \leftarrow (\mathsf{Sim}_{\mathsf{mpOT}})^{\mathcal{A}}(\sigma_1, \{\{s_{i,j,j'}\}_{j'\in\widetilde{K}_{j,i}}\}_{i\in[n]\setminus M, j\in M});\\ &\qquad\text{and } \forall i \in M, \widetilde{x}_i := \{(\widetilde{K}_{i,j}, \widetilde{s}_{i,j})\}_{j\in[n]\setminus\{i\}}\end{aligned}$$

Here, $\mathsf{View}_{\mathcal{A}}\left(\mathsf{mpOT}(\{x_i\}_{i\in[n]\setminus M})\right)$ denotes the adversary's view (including the state and transcript) and $\{\mathsf{out}_j\left(\mathsf{mpOT}(\{x_i\}_{i\in[n]\setminus M})\right)\}_{j\in[n]\setminus M}$ denote the output of honest parties in a multi-party OT protocol where honest parties have inputs $\{x_i\}_{i\in[n]\setminus M}$. Similarly, $\overline{w}^{(i)}$ denotes the transcript generated by an adversary in its interaction with a multi-party OT simulator in round i, σ_i denotes the state of the adversary at the end of the i^{th} (simulated) round, and the variable $\{\mathsf{out}_j\left(\mathcal{F}_{\mathsf{OT}}^{n\cdot(n-1)}(\{x_i\}_{i\in[n]\setminus M}, \{\widetilde{x}_i\}_{i\in M})\right)\}_{j\in[n]\setminus M}$ denote the outputs of honest parties in the ideal execution.

The simulator $\mathsf{Sim}_{\mathsf{mpOT}}$ is constructed as follows. It runs the simulator-extractor pair for the non-malleable OT, $\mathsf{Sim}_{\mathsf{OT}}, \mathsf{Ext}_{\mathsf{OT}}$, on $\mathcal{A} = \mathsf{MIM}$ to obtain:

$$\begin{aligned}&(\sigma_1, \overline{w}^{(1)}, \{\widetilde{K}_{i,j}\}_{j\in\mathbb{H}, i\in M}) \leftarrow \mathsf{Ext}_{\mathsf{OT}}^{\mathsf{MIM}}([k]^{|\mathbb{H}|\cdot|M|});\\ &(\sigma_4, \overline{w}^{(2-4)}, \{\widetilde{\mathsf{s}}_{i,j}\}_{i\in M, j\in\mathbb{H}}) \leftarrow \mathsf{Sim}_{\mathsf{OT}}^{\mathsf{MIM}, \mathsf{OT}(\{s_{i,j}\}_{i\in M, j\in\mathbb{H}}, \cdot)}(\sigma_1, \overline{w}^{(1)}, \{\widetilde{K}_{i,j}\}_{j\in\mathbb{H}, i\in M})\end{aligned}$$

We now define a sequence of hybrid experiments:

- Hyb_0: This experiment outputs the joint distribution of the view of the adversary $\mathcal{A}$ in an interaction with $\{P_i\}_{i\in[n]\setminus M}$, and the output of honest parties in this interaction, when honest parties use inputs $\{x_i = \{K_{i,j}, s_{i,j}\}_{j\in[n]\setminus\{i\}}\}_{i\in[n]\setminus M}$. The output of this experiment is the joint distribution

$$\Big(\mathsf{View}_{\mathcal{A}}\left(\mathsf{mpOT}(\{x_i\}_{i\in[n]\setminus M})\right), \{\mathsf{out}_j\left(\mathsf{mpOT}(\{x_i\}_{i\in[n]\setminus M})\right)\}_{j\in[n]\setminus M}\Big)$$

- Hyb_1 : Like in Hyb_0, we denote the input of honest parties by $\{x_i = \{K_{i,j}, s_{i,j}\}_{j\in[n]\setminus\{i\}}\}_{i\in[n]\setminus M}$. This experiment modifies the experiment in Hyb_0 to generate the view of the adversary $\mathcal{A}$ by using input $\{x_i' = \{[k], s_{i,j}\}_{j\in[n]\setminus\{i\}}\}_{i\in[n]\setminus M}$. Specifically, the honest parties' inputs used to generate the *view* of $\mathcal{A}$ are partially set to a dummy input $[k]$ instead of their real inputs $K_{i,j}$. The output of honest parties is generated identically to Hyb_0, by computing the output of the ideal watchlist functionality on input

$$(\{x_i\}_{i\in[n]\setminus M}), \{s_{i,j}\}_{j\in[n]\setminus M, i\in M})$$

  where $\{s_{i,j}\}_{j\in[n]\setminus M, i\in M}$ correspond to the implicit inputs of $\mathcal{A}$. The output of this hybrid is indistinguishable from that of Hyb_0 by stand-alone receiver security of ℓ non-malleable $\binom{m}{k}$ oblivious transfer according to Definition 5.3.

- Hyb_2 : This experiment corresponds to the joint distribution

$$\Big(\sigma_4, \overline{w}^{(1-4)}, \{\mathsf{out}_j\left(\mathcal{F}_{\mathsf{OT}}^{n\cdot(n-1)}(\{x_i\}_{i\in[n]\setminus M}, \{\widetilde{x}_i\}_{i\in M})\right)\}_{j\in[n]\setminus M}\Big)$$

where $\mathcal{A} = \mathsf{MIM}$ and

$$(\sigma_1, \overline{w}^{(1)}, \{\widetilde{K}_{i,j}\}_{j\in\mathbb{H},i\in M}) \leftarrow \mathsf{Ext}_{\mathsf{OT}}^{\mathsf{MIM}}([k]^{|\mathbb{H}|\cdot|M|});$$
$$(\sigma_4, \overline{w}^{(2-4)}, \{\widetilde{\mathsf{s}}_{i,j}\}_{i\in M,j\in\mathbb{H}}) \leftarrow \mathsf{Sim}_{\mathsf{OT}}^{\mathsf{MIM},\mathsf{OT}(\{s_{i,j}\}_{i\in M,j\in\mathbb{H}},\cdot)}(\sigma_1, \overline{w}^{(1)}, \{\widetilde{K}_{i,j}\}_{j\in\mathbb{H},i\in M})$$

The output of this hybrid is indistinguishable from that of Hyb_1 by ℓ non-malleability of $\binom{m}{k}$ OT. This follows by observing that $\mathcal{A}$ is just a man-in-the-middle adversary that participates in at most $\ell(\lambda) = n(\lambda)^2$ sessions where an honest party acts as OT sender, and at most $\ell(\lambda) = n(\lambda)^2$ sessions where an honest party acts as OT receiver, and therefore admits a simulator-extractor that on input $[k]^{|\mathbb{H}|\cdot|M|}$ produces a simulated view and extracts the implicit inputs $\{\widetilde{s}_{i,j}\}_{i\in M,j\in\mathbb{H}}$ used by $\mathcal{A}$ in this view. Therefore, the sender-security (non-malleability) property of OT according to Definition 5.3 implies that the joint distribution of the view and inputs used by $\mathcal{A}$ in Hyb_1 and Hyb_2 is computationally indistinguishable.

Note that Hyb_0 corresponds to the "real" view, and Hyb_2 to the "ideal" view, as desired. This completes the proof of security. □

6 Black-Box Five-Round Secure Multiparty Computation

In this section, we construct a five-round secure multiparty computation (MPC) protocol, and prove the following theorem.

Theorem 6.1. *Let f be any n-party functionality. Assume black-box access to the following primitives:*

- *A public-key encryption scheme with pseudorandom public keys.*
- *A two-round oblivious transfer protocol with semi-malicious security.*

Then, there exists a five-round protocol that computes f against static, malicious adversaries satisfying security with selective abort. 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 where communication is over a broadcast channel.

We use these two primitives to construct the following building blocks that will be used in our final MPC protocol.

6.1 Building Blocks

Our protocol builds on the IPS compiler [IPS08]. To instantiate this compiler, we need an *outer MPC protocol* secure against malicious adversaries that can corrupt a minority of the parties, an *inner MPC protocol* that is secure against semi-malicious adversaries that can corrupt upto all but one parties, and a watchlist protocol satisfying Definition 4.1 that securely implements the multiparty OT functionality. We now give details on each of these components and the security properties they need to satisfy.

6.1.1 Outer Protocol

As the outer protocol, we use a 2-round, n-client, m-server MPC protocol satisfying privacy with knowledge of output property (see Remark A.3) against a malicious, adaptive adversary corrupting up to $n-1$ clients and $t = (m-1)/3$ servers. Such a protocol was constructed in [IKP10, Pas12] making black-box use of a pseudorandom generator (PRG). We set $m = 8\lambda n^2$. We now give details about the syntax of this protocol.

- In the first round, the i-th client runs Φ_1 on input 1^λ, the index i and its private input x_i to obtain $(\phi_1^{i\to 1}, \ldots, \phi_1^{i\to m})$. Here, $\phi_1^{i\to j}$ denotes the private message that this client needs to send to the j-th server (for each $j \in [m]$) in the first round.
- In the second round, the j-th server runs $\Phi_2(j, (\phi_1^{1\to j}, \ldots, \phi_1^{n\to j}))$ to obtain ϕ_2^j and this is sent to the output client in the second round.
- Finally, the output client runs $\mathsf{out}_\Phi(\phi_2^1, \ldots, \phi_2^m)$ to compute the output of the protocol.

Remark 6.2. *We require the functions computed by the servers to be information-theoretic and not involve any cryptographic operations. In the protocol of [IKP10, Pas12], the servers have to perform several PRG computations. To deal with this challenge, we delegate the computation of the PRG to each of the clients. Specifically, for every PRG computation to be done by each server, every client chooses a random seed and gives the output of the PRG on this seed to the server. Let* $(\mathsf{seed}_i, \mathsf{PRG}(\mathsf{seed}_i))$ *be the contribution from the i-th client where* PRG *has a sufficiently long stretch. The server sets* $\mathsf{seed} = (\mathsf{seed}_1, \ldots, \mathsf{seed}_n)$ *and defines a new* $\mathsf{PRG}'(\mathsf{seed}) = \oplus_i \mathsf{PRG}(\mathsf{seed}_i)$. *The seed and the PRG computation is sent as part of the first message from the client to the servers. Looking ahead, we use the watchlist protocol to ensure that the PRG computations done by the honest servers are correct.*

6.1.2 Inner Protocol

To instantiate the inner protocol in the IPS compiler, we use a four-round MPC protocol that satisfies the following informal properties:

- **Rewinding Security.** Let $\mathcal{A}$ be an adversary corrupting an arbitrary subset of the parties and consider the view of $\mathcal{A}$ restricted to the first three rounds of the protocol. We require that even if (1) $\mathcal{A}$ is behaving maliciously, and (2) is able to rewind the honest parties in the first three rounds an arbitrary polynomial number of times, the view of $\mathcal{A}$ restricted to the first three rounds hides the inputs of the honest parties. We will call such an adversary as a *rewinding adversary* and denote this property as *rewinding security*.
- **Adaptive Security for the First Two Rounds.** Let $\mathcal{A}$ be a rewinding adversary corrupting an arbitrary subset of the parties. Before the third round of the protocol, the adversary $\mathcal{A}$ is allowed to send a special instruction to corrupt all the honest parties. In this case, we require that the random tape of the honest parties in the ideal experiment can be set *after execution of the first two rounds* in such a way that it is consistent with the inputs and the transcript of the first two rounds.
- **Semi-Malicious Security.** Suppose at the end of the third round, a rewinding adversary $\mathcal{A}$ produces input and randomness that explains the messages sent by all the corrupt parties in the main thread[11], then the last round message sent by the honest parties in the main thread

[11]We will call the first execution thread of the rewinding adversary as the main thread and the rest of the executions as rewinding threads

reveals no other information about their inputs except the output of the function. We call this property as *semi-malicious security*.

- **Equivocal Security.** Assume that at the end of the third round, a rewinding adversary $\mathcal{A}$ is unable to produce input and randomness that explains the messages sent by corrupted parties in the main thread. We then require the existence of an equivocal simulator that is given the inputs of all the honest parties and produces the final round message in the main thread on their behalf such that the adversary cannot distinguish whether it was interacting with the honest parties or with the simulator. We call this property as *equivocal security.*

We now give the syntax of the four-round protocol and formally define the above properties.

Syntax. The four-round inner protocol computing a function f is given by a tuple of algorithms $(\Pi_1, \ldots, \Pi_4, \mathsf{out}_\Pi)$ with the following syntax. For each round $r \in [4]$, the i-th party in the protocol runs Π_r on 1^λ, the index i, the private input x_i and the transcript of the protocol in the first $(r-1)$ rounds to obtain π_r^i. It sends π_r^i to every other party via a broadcast channel. We use $\pi(r)$ to denote the transcript of Π in the first r rounds. At the end of the interaction, parties run $\mathsf{out}_\Pi(\pi(4))$ to compute the output.[12]

Definition 6.3. *We say the protocol Π is an inner protocol for computing a funtion f 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,*
$$\Pr[\mathsf{out}_\Pi(\pi(4)) = f(x_1, \ldots, x_n)] = 1$$
where $\pi(4)$ denotes the transcript of the protocol Π when the input of P_i is x_i.

- ***Security.*** *We capture all the security properties in a real/ideal security game. 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:*
$$\textsf{\textit{Real}}(\mathcal{A}, \{x_i, r_i\}_{i \in H}) \approx_c \textsf{\textit{Ideal}}(\mathcal{A}, \mathsf{Sim}_\Pi, \{x_i\}_{i \in H})$$
where the real and ideal experiments are described in Figure 6 and for each $i \in H$, r_i is uniformly chosen.

In Section 7, we give a construction of an inner protocol from black-box use of two-round oblivious transfer with semi-malicious security.

6.1.3 Watchlist Protocol

We use a watchlist protocol $(\mathsf{WL}_1, \ldots, \mathsf{WL}_4, \mathsf{out}_{\mathsf{WL}})$ that securely implements the λ-out-of-m multiparty OT functionality (see Section 4.1) satisfying Definition 4.1.

[12]In general, the output function additionally takes in the private input and randomness of a party to generate the output. However, in our setting, the transcript of the protocol is publicly decodable, that is, the output is publicly computable given the transcript [ABG$^+$20].

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

1. For each $r \in \{1, 2\}$:
   (a) For each $i \in H$, compute $\pi_r^i := \Pi_r(1^\lambda, i, x_i, \pi(r-1); r_i)$ where $\pi(0)$ is the null string.
   (b) Send $\{\pi_r^i\}_{i \in H}$ to $\mathcal{A}$.
   (c) **Adaptive Security:** If $\mathcal{A}$ instructs to corrupt all the honest parties, then send $\{x_i, r_i\}_{i \in H}$ to $\mathcal{A}$ and output the view of $\mathcal{A}$.
   (d) **Rewinding Security:** For each $j \in [m]$ (where m is chosen by $\mathcal{A}$):
      i. Receive $\{\pi_r^i[j]\}_{i \in M}$ from $\mathcal{A}$.
      ii. For each $i \in H$, compute $\pi_{r+1}^i[j] := \Pi_{r+1}(1^\lambda, i, x_i, \pi(r)[j]; r_i)$, where $\pi(r)[j] := (\pi(r-1), \{\pi_r^i\}_{i \in H}, \{\pi_r^i[j]\}_{i \in M})$.
      iii. Send $\{\pi_{r+1}^i[j]\}_{i \in H}$ to $\mathcal{A}$.
   (e) Receive $\{\pi_r^i\}_{i \in M}$ from $\mathcal{A}$.
2. For each $i \in H$, compute $\pi_3^i := \Pi_3(1^\lambda, i, x_i, \pi(2); r_i)$.
3. Send $\{\pi_3^i\}_{i \in H}$ to $\mathcal{A}$.
4. Receive $\{\pi_3^i, (x_i, r_i)\}_{i \in M}$ from $\mathcal{A}$.
5. Check if the messages sent by corrupt parties in $\pi(3)$ are consistent with $\{x_i, r_i\}_{i \in M}$.
6. **Semi-Malicious Security:** If they are consistent:
   (a) For each $i \in H$, compute $\pi_4^i := \Pi_4(1^\lambda, i, x_i, \pi(3); r_i)$.
7. **Equivocality:** If they are not consistent:
   (a) For each $i \in H$, compute $\pi_4^i := \Pi_4(1^\lambda, i, x_i, \pi(3); r_i)$.
8. Send $\{\pi_4^i\}_{i \in H}$ to $\mathcal{A}$.
9. Receive $\{\pi_4^i\}_{i \in M}$ from $\mathcal{A}$.
10. Output the view of $\mathcal{A}$ and $\mathsf{out}_\Pi(\pi(4))$.

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

1. For each $r \in \{1, 2\}$:
   (a) For each $i \in H$, compute $\pi_r^i := \mathsf{Sim}_\Pi(1^\lambda, i, \pi(r-1))$ where $\pi(0)$ is the null string.
   (b) Send $\{\pi_r^i\}_{i \in H}$ to $\mathcal{A}$.
   (c) **Adaptive Security:** If $\mathcal{A}$ instructs to corrupt all the honest parties, then compute $\{r_i\}_{i \in H} \leftarrow \mathsf{Sim}_\Pi(\{x_i\}_{i \in H})$. Send $\{(x_i, r_i)\}_{i \in H}$ to $\mathcal{A}$ and output the view of $\mathcal{A}$.
   (d) **Rewinding Security:** For each $j \in [m]$ (where m is chosen by $\mathcal{A}$):
      i. Receive $\{\pi_r^i[j]\}_{i \in M}$ from $\mathcal{A}$.
      ii. For each $i \in H$, compute $\pi_{r+1}^i[j] := \mathsf{Sim}_\Pi(1^\lambda, i, \pi(r)[j])$, where $\pi(r)[j] := (\pi(r-1), \{\pi_r^i\}_{i \in H}, \{\pi_r^i[j]\}_{i \in M})$.
      iii. Send $\{\pi_{r+1}^i[j]\}_{i \in H}$ to $\mathcal{A}$.
   (e) Receive $\{\pi_r^i\}_{i \in M}$ from $\mathcal{A}$.
2. For each $i \in H$, compute $\pi_3^i := \mathsf{Sim}_\Pi(1^\lambda, i, \pi(2))$.
3. Send $\{\pi_3^i\}_{i \in H}$ to $\mathcal{A}$.
4. Receive $\{\pi_3^i, (x_i, r_i)\}_{i \in M}$ from $\mathcal{A}$.
5. Check if the messages sent by corrupt parties in $\pi(3)$ are consistent with $\{x_i, r_i\}_{i \in M}$.
6. **Semi-Malicious Security:** If they are consistent:
   (a) For each $i \in H$, compute $\pi_4^i \leftarrow \mathsf{Sim}_\Pi(1^\lambda, i, f(x_1, \ldots, x_n), \{x_j, r_j\}_{j \in M}, \pi(3))$.
7. **Equivocality:** If they are not consistent:
   (a) For each $i \in H$, compute $\pi_4^i \leftarrow \mathsf{Sim}_\Pi(1^\lambda, i, \{x_i\}_{i \in H}, \pi(3))$.
8. Send $\{\pi_4^i\}_{i \in H}$ to $\mathcal{A}$.
9. Receive $\{\pi_4^i\}_{i \in M}$ from $\mathcal{A}$.
10. Output the view of $\mathcal{A}$ and $\mathsf{out}_\Pi(\pi(4))$.

Figure 6: Security Game for the Inner Protocol

6.2 Construction

We now describe our five-round, black-box MPC construction in Figure 7. Let n be the number of parties and let f be the multiparty function to be securely computed. Let f' be a related functionality that takes $z_i := (\chi_i, k_i)$ from P_i where k_i is a MAC key and computes $y = f(\chi_1, \ldots, \chi_n)$ and outputs $(y, \mathsf{MAC}(k_1, y), \ldots, \mathsf{MAC}(k_n, y))$. The protocol for computing f makes use of the outer MPC protocol $(\Phi_1, \Phi_2, \mathsf{out}_\Phi)$ for securely computing the function f', m instances of the inner MPC protocol where i-th instance $\Pi_i = (\Pi_{i,1}, \ldots, \Pi_{i,4}, \mathsf{out}_{\Pi_i})$ implements the computation done by the

i-th server in the outer protocol and a watchlist protocol $(\mathsf{WL}_1, \ldots, \mathsf{WL}_4, \mathsf{out}_{\mathsf{WL}})$. We show the following theorem which implies Theorem 6.1 as a consequence of Theorem 7.1 and Theorem 5.1.

Theorem 6.4. *Let f be an arbitrary multiparty functionality. Assume black-box access to a four-round inner protocol for computing arbitrary multiparty functions (see Definition 6.3) and a four-round watchlist protocol (see Definition 4.1). The construction described in Figure 7 securely computes f in five rounds against static, malicious adversaries satisfying security with selective abort.*

6.3 Simulator

In this subsection, we give the description of the ideal world simulator Sim for our black-box MPC protocol.

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

- Sim constructs a (rushing) adversary $\mathcal{A}'$ for the first three rounds of the watchlist protocol. $\mathcal{A}'$ internally interacts with the adversary $\mathcal{A}$ as described next.
  - $\mathcal{A}'$ obtains messages on behalf of honest parties for the watchlist protocol, and generates messages on behalf of honest parties for the overall MPC protocol. Specifically, for each $i \in H$ and round $r \in [3]$, $\mathcal{A}'$ obtains watchlist messages $\{\mathsf{wl}_r^i\}_{i\in H}$.
  - It computes messages $\{\pi_{h,r-1}^i\}_{h\in[m],i\in H}$ for the inner protocol Π_h (where $\pi_{h,0}^i$ is the empty string for each $i \in H$ and $h \in [m]$) by executing the simulator Sim_{Π_h} for the inner protocol.
  - $\mathcal{A}'$ forwards $(\{\mathsf{wl}_r^i\}_{i\in H}, \{\pi_{h,r-1}^i\}_{h\in[m],i\in H})$ to $\mathcal{A}$. Next, $\mathcal{A}'$ parses the response of $\mathcal{A}$ as $\{\mathsf{wl}_r^i\}_{i\in M}, \{\pi_{h,r-1}^i\}_{h\in[m],i\in M}$, and outputs $\{\mathsf{wl}_r^i\}_{i\in M}$.
- Sim now runs $\mathsf{Sim}_{\mathsf{WL}}$ on $\mathcal{A}'$ for the first three rounds. Before sending the final (fourth) round message of WL, $\mathsf{Sim}_{\mathsf{WL}}$ queries the ideal watchlist functionality on some input, parsed as $\{x_{i,j}\}_{i\in M,j\in H}$. Sim interprets $x_{i,j}$ as a subset of $[m]$ of size λ and invokes Sim_Φ by corrupting the set of clients indexed by M and corrupting the set of servers indexed by $C := \{x_{i,j}\}_{i\in M,j\in H}$. Sim_Φ provides $\{\phi_1^{i\to j}\}_{i\in H,j\in C}$.
- For each $h \in C$, Sim instructs Sim_{Π_h} to corrupt all honest parties and gives $\{\phi_1^{i\to h}\}_{i\in H}$ as their corresponding inputs. Sim_{Π_h} provides their corresponding random tapes $\{r_{i,h}\}_{i\in H}$.
- Sim instructs $\mathcal{A}'$ to generate the third round message of the protocol Π_h on behalf of the honest parties using the randomness $\{r_{i,h}\}_{i\in H}$ and the input $\{\phi_1^{i\to h}\}_{i\in H}$ for each $h \in C$. For $h \notin C$, it instructs $\mathcal{A}'$ to generate the messages in protocol Π_h on behalf of the honest parties using the simulator Sim_{Π_h}. On obtaining the final round watchlist message, $\{\mathsf{wl}_4^i\}_{i\in H}$, $\mathcal{A}'$ runs $\mathcal{A}$ on $\{\mathsf{wl}_4^i, \{\pi_{h,3}^i\}_{h\in[m]}\}_{i\in H}$ and obtains $\{\mathsf{wl}_4^i\}_{i\in M}, \{\pi_{h,3}^i\}_{h\in[m],i\in M}$. $\mathcal{A}'$ outputs $\{\mathsf{wl}_4^i\}_{i\in M}$.
- For each $i \in H$ and $j \in M$, Sim provides $\{\phi_1^{i\to h}, r_{i,h}\}_{h\in x_{i,j}}$ to $\mathsf{Sim}_{\mathsf{WL}}$ as the response from the ideal OT between honest P_i acting as sender and corrupt P_j acting as receiver. $\mathsf{Sim}_{\mathsf{WL}}$ uses the response to generate the final round message of WL in its execution with $\mathcal{A}'$ as well as extract watchlist sender inputs $\{y_{i,j}\}_{i\in M,j\in H}$.
- For each $h \in [m]$, Sim checks if there exists some $j \in H$ such that for every $i \in M$, $y_{i,j}$ contains the input and randomness that explains the messages sent by corrupt parties in

- **Round-1:** In the first round, the party P_i with input χ_i does the following:
  1. It chooses a random MAC key $k_i \leftarrow \{0,1\}^*$ and sets $z_i := (\chi_i, k_i)$.
  2. It computes $(\phi_1^{i\to 1}, \ldots, \phi_1^{i \to m}) \leftarrow \Phi_1(1^\lambda, i, z_i)$.
  3. It chooses a random subset $K_i \subset [m]$ of size λ and sets $x_{i,j} = K_i$ for every $j \in [n] \setminus \{i\}$.
  4. It chooses a random string $r_{i,h} \leftarrow \{0,1\}^*$ for every $h \in [m]$ and sets $y_{i,j} = \{r_{i,h}, \phi_1^{i\to h}\}_{h\in[m]}$ for every $j \in [n] \setminus \{i\}$.
  5. It computes $\mathsf{wl}_1^i \leftarrow \mathsf{WL}_1(1^\lambda, i, \{x_{i,j}, y_{i,j}\}_{j\in[n]\setminus\{i\}})$.
  6. It broadcasts wl_1^i.
- **Round-2:** In the second round, P_i does the following:
  1. For each $h \in [m]$, it computes $\pi_{h,1}^i := \Pi_{h,1}(1^\lambda, i, \phi_1^{i\to h}; r_{i,h})$.
  2. It computes $\mathsf{wl}_2^i \leftarrow \mathsf{WL}_2(1^\lambda, i, \{x_{i,j}, y_{i,j}\}_{j\in[n]\setminus\{i\}}, \mathsf{wl}(1))$. (Here, $\mathsf{wl}(r)$ denotes the transcript in the first r rounds of WL.)
  3. It broadcasts $\{\pi_{h,1}^i\}_{h\in[m]}, \mathsf{wl}_2^i$.
- **Round-3:** In the third round, P_i does the following:
  1. For every $h \in [m]$, it computes $\pi_{h,2}^i := \Pi_{h,2}(1^\lambda, i, \phi_1^{i\to h}, \pi_h(1); r_{i,h})$. (Here, $\pi_h(r)$ denotes the transcript in the first r rounds of Π_h.)
  2. It computes $\mathsf{wl}_3^i \leftarrow \mathsf{WL}_3(1^\lambda, i, \{x_{i,j}, y_{i,j}\}_{j\in[n]\setminus\{i\}}, \mathsf{wl}(2))$.
  3. It broadcasts $\{\pi_{h,2}^i\}_{h\in[m]}, \mathsf{wl}_3^i$.
- **Round-4:** In the fourth round, P_i does the following:
  1. For every $h \in [m]$, it computes $\pi_{h,3}^i := \Pi_{h,3}(1^\lambda, i, \phi_1^{i\to h}, \pi_h(2); r_{i,h})$.
  2. It computes $\mathsf{wl}_4^i \leftarrow \mathsf{WL}_4(1^\lambda, i, \{x_{i,j}, y_{i,j}\}_{j\in[n]\setminus\{i\}}, \mathsf{wl}(3))$.
  3. It broadcasts $\{\pi_{h,3}^i\}_{h\in[m]}, \mathsf{wl}_4^i$.
- **Round-5:** In the fifth round, P_i does the following:
  1. It runs $\mathsf{out}_{\mathsf{WL}}$ on i, $\{x_{i,j}, y_{i,j}\}_{j\in[n]\setminus\{i\}}$, the random tape and $\mathsf{wl}(4)$ to obtain $\{r_{j,h}, \phi_1^{j\to h}\}_{j\in[n]\setminus\{i\}, h\in K_i}$.
  2. For each $j \in [n] \setminus \{i\}$ and $h \in K_i$, it checks:
     (a) If the PRG computations in $\phi_1^{j\to h}$ are correct.
     (b) For each $\ell \in [3]$, whether $\pi_{h,\ell}^j := \Pi_{h,\ell}(1^\lambda, j, \phi_1^{j\to h}, \pi_h(\ell-1); r_{j,h})$ where $\pi_h(0)$ is set to be the null string.
  3. If any of the above checks fail, then it aborts.
  4. Else, for each $h \in [m]$, it computes $\pi_{h,4}^i := \Pi_{h,4}(1^\lambda, i, \phi_1^{i\to h}, \pi_h(3); r_{i,h})$.
  5. It broadcasts $\{\pi_{h,4}^i\}_{h\in[m]}$ to every party.
- **Output Computation.** To compute the output, P_i does the following:
  1. For every $h \in [m]$, it computes $\phi_2^h := \mathsf{out}_{\Pi_h}(i, \pi_h(4))$.
  2. It computes $\mathsf{out}_\Phi(\{\phi_2^h\}_{h\in[m]})$ to recover $(y, \sigma_1, \ldots, \sigma_n)$.
  3. 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 7: Description of the Five-Round MPC Protocol

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 $(\phi_1^{i\to h}, r_{i,h})$ present in $y_{i,j}$ as the consistent input and randomness used by corrupt party P_i in the protocol Π_h.

- If $|C'| > \lambda n^2$, then Sim instructs the ideal functionality to send abort to all the honest parties and outputs the view of the adversary.

- If $|C'| \leq \lambda n^2$, then Sim instructs Sim_Φ to adaptively corrupt the servers indexed by C' and obtains $\{\phi_1^{i \to h}\}_{i \in H, h \in C'}$. For each $i \in H$, Sim chooses a random subset K_i of $[m]$ of size λ. If for any $h \in K_i$, $\{y_{j,i}\}_{j \in M}$ contains inconsistent input and randomness, then Sim instructs the ideal functionality to send abort to i. Let H' be the subset of honest parties that have not aborted.

- For every $h \in [m] \setminus \{C \cup C'\}$, Sim sends $\{\phi_1^{i \to h}\}_{i \in M}$ to Sim_Φ. Sim_Φ queries the ideal functionality f' on inputs $\{z_i := (\chi_i, k_i)\}_{i \in M}$. Sim sends $\{\chi_i\}_{i \in M}$ to its ideal functionality f and obtains y. 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 $\{\phi_2^h\}_{h \in [m] \setminus \{C \cup C'\}}$.

- To generate the final round message,
  - For each $h \in [m] \setminus \{C \cup C'\}$, Sim sends $\{(\phi_1^{i \to h}, r_{i,h})\}_{i \in M}$ as the input and the randomness of corrupt parties and ϕ_2^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 $\{\phi_1^{i \to h}\}_{i \in H}$ 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 $\{r_{i,h}, \phi_1^{i \to h}\}_{i \in H}$ to generate the final round message on behalf of $H' \subseteq H$.

- To compute the output,
  - If $H' \neq H$, then Sim instructs the ideal functionality to output abort to all the honest parties.
  - For each $h \in [m]$, Sim computes ϕ_2^h as $\mathsf{out}_{\Pi_h}(\pi_h(4))$.
  - It then computes $(y', \sigma'_1, \ldots, \sigma'_n) := \mathsf{out}_\Phi(\{\phi_2^h\}_{h \in [m]})$.
  - 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.

6.4 Proof of Indistinguishability

We now show that the real and ideal worlds are computationally indistinguishable via 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 use $\mathsf{Sim}_{\mathsf{WL}}$ to generate the messages corresponding to the watchlist protocol WL. In more detail, we construct a (rushing) adversary $\mathcal{A}'$ that plays the first four rounds of the watchlist protocol and run $\mathsf{Sim}_{\mathsf{WL}}$ on $\mathcal{A}'$ to generate the watchlist messages. $\mathcal{A}'$ internally interacts with the adversary $\mathcal{A}$ as described next.

- $\mathcal{A}'$ obtains messages on behalf of honest parties for the watchlist protocol, and generates messages on behalf of honest parties in the overall MPC protocol. Specifically, for each $i \in H$ and round $r \in [4]$, $\mathcal{A}'$ obtains watchlist messages $\{\mathsf{wl}_r^i\}_{i \in H}$. It computes messages $\{\pi_{h,r-1}^i\}_{h \in [m], i \in H}$ for the inner protocol Π_h (where $\pi_{h,0}^i$ is the empty string) by executing the same strategy as Hyb_0.
- $\mathcal{A}'$ forwards $(\{\mathsf{wl}_r^i\}_{i \in H}, \{\pi_{h,r-1}^i\}_{h \in [m], i \in H})$ to $\mathcal{A}$. Next, $\mathcal{A}'$ parses the response of $\mathcal{A}$ as $\{\mathsf{wl}_r^i\}_{i \in M}, \{\pi_{h,r-1}^i\}_{h \in [m], i \in M}$, and outputs $\{\mathsf{wl}_r^i\}_{i \in M}$.

Note that before sending the final (fourth) round message of WL, $\mathsf{Sim}_{\mathsf{WL}}$ queries the ideal watchlist functionality, which is answered using the honest party inputs $\{x_{i,j}, y_{i,j}\}_{i \in H, j \in [n] \setminus \{i\}}$.

We show that Hyb_0 and Hyb_1 are computationally indistinguishable in Claim 6.5 by relying on the security of the watchlist protocol.

- $\underline{\mathsf{Hyb}_2 :}$ This is the same as the previous hybrid, except that we instruct $\mathcal{A}'$ to generate the protocol messages for $\{\Pi_h\}_{h \in [m]}$ as described in the simulation.

  We show that Hyb_1 is computationally indistinguishable from Hyb_2 using the security of the inner protocol in Claim 6.6.

- $\underline{\mathsf{Hyb}_3 :}$ In this hybrid, we define the set C' as in the simulation strategy, and if C' has size greater than λn^2, we instruct the honest parties to abort.

  We show that Hyb_2 is statistically close to Hyb_3 in Claim 6.7.

- $\underline{\mathsf{Hyb}_4}$: 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 show that Hyb_3 is computationally indistinguishable from Hyb_4 by the security of the outer protocol in Claim 6.8.

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

  - When Sim_Φ queries the ideal functionality f' on $\{\chi_i, k_i\}_{i \in M}$, we query f on $\{\chi_i\}_{i \in M}$ 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.

  Note that Hyb_5 is identically distributed to the ideal execution. We show in Claim 6.9 that Hyb_4 is statistically close to Hyb_5 from the security of the MAC scheme.

Claim 6.5. *Assuming the security of the watchlist protocol* WL, *we have that* $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$.

Proof. Assume for the sake of contradiction that there exists a distinguisher D that distinguishes Hyb_0 from Hyb_1 with non-negligible advantage. We will use this distinguisher to construct an adversary $\mathcal{B}$ that breaks the security of the watchlist protocol WL.

For each $i \in H$, $\mathcal{B}$ chooses a random subset K_i of $[m]$ of size λ and sets $x_{i,j} := K_i$ for each $j \in [n] \setminus \{i\}$. $\mathcal{B}$ computes $\Phi_1(1^\lambda, i, z_i)$ to obtain $(\phi_1^{i \to 1}, \ldots, \phi_1^{i \to m})$ for each $i \in H$. It then chooses a random string $r_{i,h} \leftarrow \{0,1\}^*$ and sets $y_{i,j} = \{(r_{i,h}, \phi_1^{i \to h})\}_{h \in [m]}$ for each $j \in [n] \setminus \{i\}$. $\mathcal{B}$ sends $\{(x_{i,j}, y_{i,j})\}_{i \in H, j \in [n] \setminus \{i\}}$ as the inputs of the honest parties in the watchlist protocol to the external challenger. $\mathcal{B}$ interacts with an external challenger to send and receive the messages corresponding to the watchlist protocol. $\mathcal{B}$ constructs the adversary $\mathcal{A}'$ as described in Hyb_1 and forwards the watchlist protocol messages received from challenger to $\mathcal{A}'$ and in turn forwards the watchlist messages output by $\mathcal{A}'$ to the challenger. At the end of the fourth round, $\mathcal{B}$ receives the output of the watchlist given to the honest parties that comprises of $\{r_{j,h}, \phi_1^{j \to h}\}_{i \in H, j \in [n] \setminus \{i\}, h \in K_i}$. It performs the same checks that an honest party would perform before the fifth round message and computes the outputs of all the honest parties. Finally, $\mathcal{B}$ runs the distinguisher D on the view of the adversary together with the outputs of the honest parties and mirrors the output of D.

Note that if watchlist protocol messages are simulated, then the input to D is distributed identically to Hyb_1, and otherwise to Hyb_0. Thus, if D is able to distinguish between Hyb_0 and Hyb_1 with non-negligible advantage, then $\mathcal{B}$ can break the security of the watchlist protocol which is a contradiction. □

Claim 6.6. *Assuming the security of the inner protocol, we have that* $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$.

Proof. 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 $h \in [m]$ 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 $k < h$, the messages in the protocol Π_k are generated using the simulator Sim_{Π_k} and for every $k > h$, 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 $\mathsf{Hyb}_{1,m}$ is distributed identically to Hyb_2. We now construct an adversary $\mathcal{B}$ that uses D and breaks security of the inner protocol.

$\mathcal{B}$ computes $\Phi_1(1^\lambda, i, z_i)$ to obtain $(\phi_1^{i \to 1}, \ldots, \phi_1^{i \to m})$ for each $i \in H$. It sends $\{\phi_1^{i \to h}\}_{i \in H}$ as the inputs of the honest parties in Π_h to the external challenger.

Next, $\mathcal{B}$ constructs $\mathcal{A}'$ which internally interacts with the adversary $\mathcal{A}$ in the first three rounds, as described below.

- $\mathcal{A}'$ obtains as input – messages on behalf of all the honest parties for the watchlist protocol, and generates as output – messages on behalf of honest parties for the 5-round MPC. Specifically, for each $i \in H$ and round $r \in [3]$, $\mathcal{A}'$ obtains watchlist messages $\{\mathsf{wl}_r^i\}_{i \in H}$. It computes messages $\{\pi_{h',r-1}^i\}_{h' \in [m], i \in H}$ for the inner protocol by executing the strategy in $\mathsf{Hyb}_{1,h-1}$, but with the following change. For each $h' \neq h$ and $i \in H$, $\mathcal{A}'$ generates the protocol messages in $\pi_{h',r-1}^i$ as in $\mathsf{Hyb}_{1,h-1}$. To generate the protocol messages for Π_h, $\mathcal{B}$ interacts with the external challenger and asks $\mathcal{A}'$ to embed the messages obtained from the external challenger as the protocol messages corresponding to $\pi_{h,r-1}^i$ for every $r \in [3]$ and $i \in H$.

- $\mathcal{A}'$ forwards $(\{\mathsf{wl}_r^i\}_{i \in H}, \{\pi_{h',r-1}^i\}_{h' \in [m], i \in H})$ to $\mathcal{A}$. It parses the response of $\mathcal{A}$ as $\{\mathsf{wl}_r^i\}_{i \in M}$, $\{\pi_{h',r-1}^i\}_{h' \in [m], i \in M}$, and outputs $\{\mathsf{wl}_r^i\}_{i \in M}$. $\mathcal{B}$ forwards $\{\pi_{h,r-1}^i\}_{i \in M}$ to the external challenger.

$\mathcal{B}$ runs $\mathsf{Sim}_{\mathsf{WL}}$ on $\mathcal{A}'$. Note that before sending the final (fourth) round message of WL, $\mathsf{Sim}_{\mathsf{WL}}$ queries the ideal watchlist functionality, which is provided honest party inputs $\{x_{i,j}, y_{i,j}\}_{i \in H, j \in [n] \setminus \{i\}}$. $\mathcal{B}$ parses the simulator's query to extract $\{x_{i,j}\}_{i \in M, j \in H}$. $\mathcal{B}$ interprets $x_{i,j}$ as a subset of $[m]$ of size λ and sets $C = \{x_{i,j}\}_{i \in M, j \in H}$.

- If $h \in C$, $\mathcal{B}$ instructs the external challenger to corrupt all the honest parties in Π_h and obtains $\{r_{i,h}\}_{i \in H}$. $\mathcal{B}$ instructs $\mathcal{A}'$ to use $\{r_{i,h}, \phi_1^{i \to h}\}_{i \in H}$ to compute the third round protocol message of Π_h.

- If $h \notin C$, $\mathcal{B}$ instructs $\mathcal{A}'$ to continue its interaction with $\mathcal{A}$ until the end of the fourth round by obtaining third round messages from the external challenger for Π_h.

After completion of the fourth round, $\mathsf{Sim}_{\mathsf{WL}}$ extracts $\{y_{i,j}\}_{i \in M, j \in H}$. $\mathcal{B}$ now checks if there exists some $j \in H$ such that for every $i \in M$, $y_{i,j}$ 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 such a j exists, then $\mathcal{B}$ extracts $(r_{i,h}, \phi_1^{i \to h})$ from $y_{i,j}$ and sets it as the consistent input and randomness used by corrupt party P_i in the protocol Π_h. It sends this to the external challenger. On the other hand, if such a j does not exist, then $\mathcal{B}$ sends some dummy input and dummy randomness to the external challenger. $\mathcal{B}$ receives the final round message in protocol Π_h and uses the final round message from $\mathcal{A}$ to compute the output of Π_h. All other protocol messages are generated exactly as in $\mathsf{Hyb}_{1,h-1}$.

Finally, $\mathcal{B}$ runs the distinguisher D on the view of the adversary and the outputs of the honest parties. Note that if the messages generated by the external challenger are distributed identically to the real execution of the protocol, then the inputs to D are distributed identically to $\mathsf{Hyb}_{1,h-1}$. Otherwise, they are distributed identically to $\mathsf{Hyb}_{1,h}$. Since D can distinguish between $\mathsf{Hyb}_{1,h-1}$ and $\mathsf{Hyb}_{1,h}$ with non-negligible advantage, this means that $\mathcal{B}$ can break the security of the inner protocol which is a contradiction. □

Claim 6.7. $\mathsf{Hyb}_2 \approx_s \mathsf{Hyb}_3$.

Proof. Fix any honest party P_i. Note that P_i aborts in Hyb_2 if $|K_i \cap C'| \neq 0$. We show that if $|C'| > \lambda n^2$ then the probability of $|K_i \cap C'| = 0$ is negligible.

Note that K_i is distributed as a random subset of $[m]$ of size λ. We now upper bound the probability that $|K_i \cap C'| = 0$.

$$\begin{aligned}
\Pr[|K_i' \cap C'| = 0] &= \frac{\binom{m-|C'|}{\lambda}}{\binom{m}{\lambda}} \\
&< \frac{\binom{m-\lambda n^2}{\lambda}}{\binom{m}{\lambda}} \\
&= \frac{(m-\lambda n^2)!}{m!} \frac{(m-\lambda)!}{(m-\lambda-\lambda n^2)!} \\
&< (1-\lambda/m)^{\lambda n^2} \\
&< 2^{-O(\lambda)}
\end{aligned}$$

The claim now follows from a standard union bound over the set of all honest parties. □

Claim 6.8. *Assuming the privacy with knowledge of outputs property of the outer MPC protocol, we have* $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$.

Proof. 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.

$\mathcal{B}$ chooses a random MAC key k_i for each $i \in H$ and sends $\{z_i := (\chi_i, k_i)\}_{i \in H}$ as the inputs of the honest clients in the outer MPC protocol to the external challenger. $\mathcal{B}$ constructs $\mathcal{A}'$ that interacts internally with $\mathcal{A}$ in the first three rounds as in Hyb_3.

Before sending the final round message of WL, $\mathsf{Sim}_{\mathsf{WL}}$ queries the ideal watchlist functionality on input $\{x_{i,j}\}_{i \in M, j \in H}$. $\mathcal{B}$ interprets $x_{i,j}$ as a subset of $[m]$ of size λ and instructs the external challenger to adaptively corrupt the set of servers indexed by $C := \{x_{i,j}\}_{i \in M, j \in H}$.

The challenger provides $\{\phi_1^{i\to j}\}_{i\in H, j\in C}$. $\mathcal{B}$ uses this as input to Sim_{Π_h} for each $h \in C$. At the end of the fourth round, $\mathcal{B}$ constructs the set C' as in Hyb_3. If $|C'| < \lambda n^2$, then $\mathcal{B}$ instructs the external challenger to corrupt the servers indexed by C' and obtains $\{\phi_1^{i\to h}\}_{i\in H, h\in C'}$. For every $h \in [m] \setminus \{C \cup C'\}$, $\mathcal{B}$ sends $\{\phi_1^{i\to h}\}_{i\in M}$ as the first round messages generated by corrupted client to the honest server indexed by h. $\mathcal{B}$ obtains $\{\phi_2^h\}_{h\in[m]\setminus\{C\cup C'\}}$. $\mathcal{B}$ uses this to generate the final round message of the protocol.

On receiving the final round message from $\mathcal{A}$, $\mathcal{B}$ computes $\{\phi_2^h\}_{h\in[m]}$ using out_{Π_h} and sends $\{\phi_2^h\}_{h\in C\cup C'}$ to the external challenger as the second round message. It runs out_Φ on the second round messages and computes $(y', \sigma_1', \ldots, \sigma_n')$. It then performs the MAC checks as in the description of the protocol. $\mathcal{B}$ runs D on the view of the adversary and the outputs of the honest parties and outputs whatever D outputs.

Since $|C| < \lambda n^2$ and $|C'| \leq \lambda n^2$, the size of $|C \cup C'| < 2\lambda n^2 < (m-1)/3$. 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. □

Claim 6.9. *Assuming the security of the MAC scheme, we have* $\mathsf{Hyb}_4 \approx_s \mathsf{Hyb}_5$.

Proof. Note that it follows from the property of the tags that for any y, $\mathsf{MAC}(k, y)$ is uniformly distributed for a randomly chosen MAC key k. Therefore, for each $i \in H$, σ_i computed in both Hyb_4 and in Hyb_5 are distributed identically. Thus, the only difference between these two hybrids is that in Hyb_5, we check if $y = y'$ and for each $i \in H$, we check if $\sigma_i' = \sigma_i$. For the honest parties where this check fails, we instruct them to abort. For all other honest parties, we instruct them to output y. On the other hand, in Hyb_4, we instruct the honest parties to do the MAC verification and depending on the result of the verification procedure, they abort or output y'. If an honest party does not abort in Hyb_5, then it follows from the correctness of the verification procedure that it does not abort in Hyb_4. Suppose there exists an honest party P_i that aborts in Hyb_5, but with non-negligible probability does not abort in Hyb_4. Then, this means that $(y', \sigma_i') \neq (y, \sigma_i)$ and that the verification procedure on (y', σ_i') outputs 1. However, this contradicts the security of the MAC scheme. □

7 Inner Protocol

In this section, we construct a four-round inner protocol satisfying the properties described in Section 6.1.2. This protocol makes black-box use of a two-round oblivious transfer with security against semi-malicious adversaries. The main theorem we show is the following:

Theorem 7.1. *Let f be an arbitrary multiparty functionality. Assume black-box access to a two-round semi-malicious oblivious transfer. There exists an inner protocol Π that computes f satisfying Definition 6.3.*

Our inner protocol builds on the round-collapsing compiler given in [GS18, BL18]. We first recall the notion of conforming protocols introduced in [GS18] which is one of the key components in constructing the round-collapsing compiler.

7.1 Conforming Protocols

In this subsection, we recall the definition of conforming protocols from [GS18, GIS18]. We follow the convention given in [GIS18] and most parts of this subsection are taken verbatim from [GIS18].

Specifications for a Conforming Protocol. Consider an n-party deterministic[13] MPC protocol Φ between parties $P_1, \ldots, P_n$ that computes a functionality f. For each $i \in [n]$, we let $x_i \in \{0,1\}^m$ denote the input of party P_i. A conforming protocol Φ is defined by functions pre, post, and computations steps or what we call *actions* $\phi_1, \cdots \phi_T$. The protocol Φ proceeds in three stages: the pre-processing stage, the computation stage, and the output stage.

- **Pre-processing phase**: For each $i \in [n]$, party P_i computes

  $$(z_i, v_i) \leftarrow \mathsf{pre}(1^\lambda, i, x_i)$$

  where pre is a randomized algorithm. The algorithm pre takes as input the index i of the party, its input x_i and outputs $z_i \in \{0,1\}^{\ell/n}$ and $v_i \in \{0,1\}^\ell$ (where ℓ is a parameter of the protocol). Finally, P_i retains v_i as the secret information and broadcasts z_i to every other party.

- **Computation phase**: For each $i \in [n]$, party P_i sets

  $$\mathsf{st} := (z_1 \| \cdots \| z_n)$$

  Next, for each $t \in \{1 \cdots T\}$ parties proceed as follows:

  1. Parse action ϕ_t as (i, a, b, c) where $i \in [n]$ and $a, b, c \in [\ell]$.
  2. Party P_i computes *one* NAND gate as

     $$\mathsf{st}_c = \mathsf{NAND}(\mathsf{st}_a \oplus v_{i,a}, \mathsf{st}_b \oplus v_{i,b}) \oplus v_{i,c}$$

     and broadcasts st_c to every other party.
  3. Every party P_j for $j \neq i$ updates st_c to the bit value received from P_i.

  We require that for all $t, t' \in [T]$ such that $t \neq t'$, we have that if $\phi_t = (\cdot,\cdot,\cdot,h)$ and $\phi_{t'} = (\cdot,\cdot,\cdot,h')$ then $h \neq h'$. Also, we denote $A_i \subset [T]$ to be the set of rounds in with party P_i sends a bit. Namely, $A_i = \{t \in T \mid \phi_t = (i,\cdot,\cdot,\cdot)\}$.

- **Output phase**: For each $i \in [n]$, party P_i outputs $\mathsf{post}(i, \mathsf{st})$.

Lemma 7.2 ([GS18, GIS18]). *Any MPC protocol can be transformed to a conforming protocol Φ with a polynomial overhead while inheriting the correctness and the security of the original protocol.*

Observing that the protocols in [GMW87, Kil88, IPS08] have information-theoretic, semi-malicious security in the presence of OT-correlations between every pair of parties, we get the following corollary.

Corollary 7.3. *For any n-party functionality represented as a circuit C, there exists a perfectly secure conforming protocol Φ against semi-malicious adversaries for computing C in the presence $O(|C|)$ OT-correlation tuples between every pair of parties.*

We now observe the following property about the proof of Lemma 7.2 from [GS18, GIS18].

Property 7.4. *Let Φ be a conforming protocol with security against semi-malicious adversaries. Let Sim_Φ be the corresponding simulator. There exists a special algorithm Sim_Φ^{eq} such that for every $x_i \in \{0,1\}^m$, the following two distributions are identical.*

- $z_i \leftarrow \mathsf{Sim}_\Phi(1^\lambda, i)$ *and* $v_i \leftarrow \mathsf{Sim}_\Phi^{eq}(i, x_i, z_i)$. *Output* (z_i, v_i).
- $(z_i, v_i) \leftarrow \mathsf{pre}(1^\lambda, i, x_i)$. *Output* (z_i, v_i).

[13]Randomized protocols can be handled by including the randomness used by a party as part of its input.

7.2 Special Two-Round Oblivious Transfer

In this subsection, we construct a special two-round oblivious transfer protocol. We start with the syntax and then give the security properties to be satisfied by the OT protocol.

Syntax. Let $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ be a two-round oblivious transfer with the following syntax. OT_1 takes the security parameter 1^λ and the receiver's choice bit and outputs the first round message otm_1 along with some secret state ω. OT_2 takes otm_1 and the two sender inputs m_0 and m_1 and outputs otm_2. OT_3 takes otm_2 and (b, ω) and outputs m_b. We require the OT protocol to satisfy the following properties:

- **Correctness:** For every input b of the receiver and m_0, m_1 of the sender:

$$\Pr[\mathsf{OT}_3(\mathsf{otm}_2, (b, \omega)) = m_b] = 1$$

  where $(\mathsf{otm}_1, \omega) \leftarrow \mathsf{OT}_1(1^\lambda, b)$ and $\mathsf{otm}_2 \leftarrow \mathsf{OT}_2(\mathsf{otm}_1, m_0, m_1)$.

- **Equivocal Receiver Security.** There exists a special algorithm $\mathsf{Sim}^{eq}_{\mathsf{OT}}$ that on input 1^λ outputs $(\mathsf{otm}_1, \omega_0, \omega_1)$ such that for any $b \in \{0, 1\}$,

$$\{(\mathsf{otm}_1, \omega_b) : (\mathsf{otm}_1, \omega_0, \omega_1) \leftarrow \mathsf{Sim}^{eq}_{\mathsf{OT}}(1^\lambda)\} \approx_c \{(\mathsf{otm}_1, \omega) : (\mathsf{otm}_1, \omega) \leftarrow \mathsf{OT}_1(1^\lambda, b)\}$$

- **Security in the No Corruption Setting.** For any two bits b, b' and two sets of sender inputs (m_0, m_1) and (m'_0, m'_1), we have:

$$\{(\mathsf{otm}_1, \mathsf{otm}_2) : (\mathsf{otm}_1, \omega) \leftarrow \mathsf{OT}_1(1^\lambda, b), \mathsf{otm}_2 \leftarrow \mathsf{OT}_2(\mathsf{otm}_1, m_0, m_1)\} \approx_c$$
$$\{(\mathsf{otm}_1, \mathsf{otm}_2) : (\mathsf{otm}_1, \omega) \leftarrow \mathsf{OT}_1(1^\lambda, b'), \mathsf{otm}_2 \leftarrow \mathsf{OT}_2(\mathsf{otm}_1, m'_0, m'_1)\}$$

- **Sender Privacy:** For any input m_0, m_1 of the sender and any bit b and a string $r \in \{0, 1\}^*$:

$$\{b, r, \mathsf{otm}_1 := \mathsf{OT}_1(1^\lambda, b; r), \mathsf{OT}_2(\mathsf{otm}_1, m_0, m_1)\} \approx_c \{b, r, \mathsf{otm}_1 := \mathsf{OT}_1(1^\lambda, b; r), \mathsf{OT}_2(\mathsf{otm}_1, m_b, m_b)\}$$

In Appendix B, we show a construction of such a two-round oblivious transfer protocol that makes black-box use of a two-round oblivious transfer against semi-malicious adversary.

7.3 Construction

Let n be the number of parties and let f be the multiparty functionality to be securely computed. We describe the inner protocol computing f in Figure 8. This protocol makes use of the following building blocks:

- A perfectly secure conforming protocol Φ for computing f against semi-malicious adversaries in the OT correlations model. Let us assume that the number of correlations required between each ordered pair of parties is m.
- A two-round special oblivious transfer protocol from Section 7.2.
- A pseudorandom function PRF.

- For each round $r \in [4]$, each P_i chooses a random PRF key $k_r^i \leftarrow \{0,1\}^*$ and the random bits used in computing the protocol messages in r-th round are derived by applying the PRF on the transcript seen so far.
- **Rounds-1 & 2:** In the first two rounds, for each $i, j \in [n] \times [n]$ such that $i \neq j$, P_i acts a receiver in m special OT invocations with sender P_j where the inputs of P_i and P_j in each of these m invocations are (pseudo)randomly chosen (using the PRF).
- **Round-3:** Each P_i does the following:
  1. Let y_i denote the augmented input of P_i that includes the actual input x_i as well as the OT correlations generated in the first two rounds.
  2. It computes $(z_i, v_i) \leftarrow \mathsf{pre}(i, y_i)$.
  3. For each $t \in A_i$ and $\alpha, \beta \in \{0,1\}$,
     (a) Let $\phi_t = (i, (a, b, c))$.
     (b) It computes $r^{t,\alpha,\beta} = (v_{i,c} \oplus \mathsf{NAND}(v_{i,a} \oplus \alpha, v_{i,b} \oplus \beta))$.
     (c) It computes $(\mathsf{otm}_1^{t,\alpha,\beta}, \omega^{t,\alpha,\beta}) \leftarrow \mathsf{OT}_1(1^\lambda, r^{t,\alpha,\beta})$.
  4. It sends $\left(z_i, \{\mathsf{otm}_1^{t,\alpha,\beta}\}_{t \in A_i, \alpha,\beta \in \{0,1\}}\right)$ to every other party.
- **Round-4:** In the final round, each P_i does the following:
  1. It sets $\mathsf{st} := (z_1 \| \ldots \| z_i \| \ldots \| z_n)$.
  2. It sets $\mathsf{lab}^{i,T+1} := \{\mathsf{lab}_{k,0}^{i,T+1}, \mathsf{lab}_{k,1}^{i,T+1}\}_{k \in [\ell]}$ where for each $k \in [\ell]$ and $b \in \{0,1\}$, $\mathsf{lab}_{k,b}^{i,T+1} := 0^\lambda$.
  3. For each t from T down to 1,
     (a) It parses ϕ_t as $(i^*, (a, b, c))$.
     (b) If $i = i^*$, it computes $\left(\widetilde{f}^{i,t}, \mathsf{lab}^{i,t}\right) \leftarrow \mathsf{Garble}(1^\lambda, f^{i,t}[v_i, \{\omega^{t,\alpha,\beta}\}_{\alpha,\beta}, \perp, \mathsf{lab}^{i,t+1}])$. where $f^{i,t}$ is described in Figure 9.
     (c) If $i \neq i^*$,
        i. For every $\alpha, \beta \in \{0,1\}$, it computes $\mathsf{otm}_{2,i}^{t,\alpha,\beta} \leftarrow \mathsf{OT}_2(\mathsf{otm}_1^{t,\alpha,\beta}, \mathsf{lab}_{c,0}^{i,t+1}, \mathsf{lab}_{c,1}^{i,t+1})$.
        ii. It computes $\left(\widetilde{f}^{i,t}, \mathsf{lab}^{i,t}\right) \leftarrow \mathsf{Garble}(1^\lambda, f^{i,t}[v_i, \perp, \{\mathsf{otm}_{2,i}^{t,\alpha,\beta}\}_{\alpha,\beta}, \mathsf{lab}^{i,t+1}])$. where $f^{i,t}$ is described in Figure 9.
  4. It sends $\left(\{\widetilde{f}^{i,t}\}_{t \in [T]}, \{\mathsf{lab}_{k,\mathsf{st}_k}^{i,1}\}_{k \in [\ell]}\right)$ to every other party.
- **Output:** To compute the output, P_i does the following:
  1. For each $k \in [n]$, let $\widetilde{\mathsf{lab}}^{k,1}$ be the input labels received from P_k at the end of round 2.
  2. For each t from 1 to T do:
     (a) It parses ϕ_t as $(i^*, (a, b, c))$.
     (b) It computes $((\gamma, \omega), \widetilde{\mathsf{lab}}^{i^*,t+1}) := \mathsf{Eval}(\widetilde{f}^{i^*,t}, \widetilde{\mathsf{lab}}^{i^*,t})$.
     (c) It updates $\mathsf{st}_c := \gamma$.
     (d) For each $k \neq i^*$ do:
        i. It computes $(\mathsf{otm}_{2,i}, \{\mathsf{lab}_h^{k,t+1}\}_{h \in [\ell] \setminus \{c\}}) := \mathsf{Eval}(\widetilde{f}^{i,t}, \widetilde{\mathsf{lab}}^{i,t})$.
        ii. It recovers $\mathsf{lab}_c^{k,t+1} := \mathsf{OT}_3(\mathsf{otm}_{2,i}, (\gamma, \omega))$.
        iii. It sets $\widetilde{\mathsf{lab}}^{k,t+1} := \{\mathsf{lab}_h^{k,t+1}\}_{h \in [\ell]}$.
  3. Compute the output as $\mathsf{post}(i, \mathsf{st})$.

Figure 8: Inner Protocol based on [GIS18]

$f^{i,t}$

Input. st.
Hardcoded. v_i, the strings $\{\omega^{t,\alpha,\beta}\}_{\alpha,\beta}$, $\{\mathsf{ots}_{2,i}^{t,\alpha,\beta}\}_{\alpha,\beta}$ and a set of labels $\mathsf{lab} = \{\mathsf{lab}_{k,0}, \mathsf{lab}_{k,1}\}_{k\in[\ell]}$.

1. Let $\phi_t = (i^*, (a, b, c))$.
2. If $i = i^*$ then:
   (a) Compute $\mathsf{st}_c := \mathsf{NAND}(\mathsf{st}_a \oplus v_{i,a}, \mathsf{st}_b \oplus v_{i,b}) \oplus v_{i,c}$ and $\gamma := \mathsf{st}_c$.
   (b) Output $(\gamma, \omega^{t,\mathsf{st}_a,\mathsf{st}_b}, \{\mathsf{lab}_{k,\mathsf{st}_k}\}_{k\in[\ell]})$.
3. Else:
   (a) Output $(\mathsf{ots}_2^{t,\mathsf{st}_a,\mathsf{st}_b}, \{\mathsf{lab}_{k,\mathsf{st}_k}\}_{k\in[\ell]\setminus\{c\}})$.

Figure 9: The circuit $f^{i,t}$.

7.4 Simulator

Let $\mathcal{A}$ be an adversary corrupting a subset of parties indexed by the set M and let $H := [m] \setminus M$. We now give the description of the ideal world simulator Sim_Π.

- For each $i \in H$ and each round $r \in [4]$, Sim_Π chooses a random PRF key $k_r^i \leftarrow \{0,1\}^*$ and derives the randomness for generating the messages on behalf of P_i in round-r by applying the PRF on the transcript seen so far.
- **Rounds 1 & 2:** For each $i \in H$, Sim_Π generates the round-1 and round-2 messages on behalf of the honest party P_i exactly as described in the protocol. If the adversary requests to corrupt all the honest parties, then Sim_Π sets $r_i := (k_1^i, \ldots, k_4^i)$ for each $i \in h$.
- **Round-3:** To generate the round-3 message on behalf of honest P_i, Sim_Π does the following:
  - It samples $z_i \leftarrow \mathsf{Sim}_\Phi(i)$.
  - For each $t \in A_i$, $\alpha, \beta \in \{0,1\}$,
    - For each $b \in \{0,1\}$, it samples $(\mathsf{otm}_1^{t,\alpha,\beta}, \omega_0, \omega_1) \leftarrow \mathsf{Sim}_{\mathsf{OT}}^{eq}$ and sets $\overline{\omega}^{t,\alpha,\beta} := (\omega_0, \omega_1)$.
  - It sends $\left(z_i, \{\mathsf{otm}_1^{t,\alpha,\beta}\}_{t\in A_i,\alpha,\beta\in\{0,1\}}\right)$ to $\mathcal{A}$.
- **Round-4:** To generate the round-4 message on behalf of honest P_i, Sim_Π does the following:
  - **Case-1:** $\{x_i, r_i\}_{i\in M}$ **is inconsistent.** In this case, Sim_Π receives $\{x_i\}_{i\in H}$ and:
    1. It sets y_i to be the augmented input of P_i that includes the actual input x_i as well as the OT correlations generated in the first two rounds.
    2. It computes $v_i \leftarrow \mathsf{Sim}_\Phi^{eq}(i, z_i, y_i)$.
    3. For each $t \in A_i$, $\alpha, \beta \in \{0,1\}$, it computes $r^{t,\alpha,\beta} = (v_{i,c} \oplus \mathsf{NAND}(v_{i,a} \oplus \alpha, v_{i,b} \oplus \beta))$. It parses $\overline{\omega}^{t,\alpha,\beta}$ as (ω_0, ω_1) and sets $\omega^{t,\alpha,\beta} = \omega_{r^{t,\alpha,\beta}}$.
    4. It generates the round-4 message on behalf of P_i exactly as described in the protocol using v_i, $\{\omega^{t,\alpha,\beta}\}_{t\in A_i,\alpha,\beta\in\{0,1\}}$.

– **Case-2:** $\{x_i, r_i\}_{i \in M}$ **is consistent.** In this case, Sim_Π receives $\{x_i, r_i\}_{i \in M}, f(x_1, \ldots, x_n)$ and:

  1. For each $i \in M$, it recovers y_i from x_i, r_i and the transcript in the first three rounds and runs Sim_Φ on input $\{y_i\}_{i \in M}$ and $f(x_1, \ldots, x_n)$ and obtains the transcript Z of the computation phase of the protocol. Let st^* be the final state of the parties updated with the transcript Z.
  2. For each $k \in [\ell]$, it sets $\mathsf{lab}_k^{i,T+1} := 0^\lambda$.
  3. For each t from T down to 1,
     (a) It parses ϕ_t as (i^*, a, b, c).
     (b) It sets $\alpha^* := \mathsf{st}_a^*$, $\beta^* := \mathsf{st}_b^*$, and $\gamma^* := \mathsf{st}_c^*$.
     (c) If $i = i^*$ then it parses $\overline{\omega}^{t,\alpha^*,\beta^*}$ as (ω_0, ω_1) and sets $\omega^{t,\alpha^*,\beta^*} = \omega_{\gamma^*}$. It computes

$$\left(\widetilde{f}^{i,t}, \{\mathsf{lab}_k^{i,t}\}_{k \in [\ell]}\right) \leftarrow \mathsf{Sim}_{\mathsf{Garb}}\left(1^{|f_{i,t}|}, 1^\ell, \left(\gamma^*, \omega^{t,\alpha^*,\beta^*}, \{\mathsf{lab}_k^{i,t+1}\}_{k \in [\ell]}\right)\right).$$

     (d) Else, it computes $\mathsf{otm}_{2,i}^{t,\alpha^*,\beta^*} \leftarrow \mathsf{OT}_2(\mathsf{otm}_1^{t,\alpha^*,\beta^*}, \mathsf{lab}_c^{i,t+1}, \mathsf{lab}_c^{i,t+1})$. It computes

$$\left(\widetilde{f}^{i,t}, \{\mathsf{lab}_k^{i,t}\}_{k \in [\ell]}\right) \leftarrow \mathsf{Sim}_{\mathsf{Garb}}\left(1^{|f_{i,t}|}, 1^\ell, \left(\mathsf{otm}_{2,i}^{t,\alpha^*,\beta^*}, \{\mathsf{lab}_k^{i,t+1}\}_{k \in [\ell] \setminus \{c\}}\right)\right).$$

– It sends $\left\{\left(\{\widetilde{f}^{i,t}\}_{t \in [T]}, \{\mathsf{lab}_k^{i,1}\}_{k \in [\ell]}\right)\right\}_{i \in H}$ to $\mathcal{A}$.

7.5 Proof of Indistinguishability

In this subsection, we show that $\mathsf{Real}(\mathcal{A}, \{x_i, r_i\}_{i \in H}) \approx_c \mathsf{Ideal}(\mathcal{A}, \mathsf{Sim}_\Pi, \{x_i\}_{i \in H})$ via a hybrid argument.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the output of $\mathsf{Real}(\mathcal{A}, \{x_i, r_i\}_{i \in H})$.
- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we make the following changes:
  – Set $\mathsf{count} = 0$.
  – While ($\mathsf{count} \leq \lambda$)
    * Choose a random bit $\mathsf{guess} \leftarrow \{0, 1\}$.
    * Interact with the adversary in the first two rounds as specified in the experiment Real.
    * Let χ be the indicator bit that is 1 iff the adversary sends the instruction to corrupt all the honest parties.
    * If $\chi = \mathsf{guess}$, then exit the loop and proceed to the subsequent steps.
    * If $\chi \neq \mathsf{guess}$, then increment count and go back to the first step of this while loop.
  – If $\mathsf{count} = \lambda + 1$, then abort the experiment and output a special symbol fail.

  In Claim 7.5, we show that Hyb_0 is statistically indistinguishable to Hyb_1.
- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we make the following changes in every execution of the while loop:
  – Choose a random bit $\mathsf{guess} \leftarrow \{0, 1\}$.
  – If ($\mathsf{guess} = 1$), then interact with the adversary in the first two rounds as in Hyb_1.

- If ($\mathsf{guess} = 0$), then for each $i \in H$ and each round $r \in [4]$, we derive the randomness used in generating the protocol messages in round-r by applying a random function on the transcript instead of deriving them using the PRF.
- The rest of the steps are identical to Hyb_1.

In Claim 7.6, we show that $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$ from the security of the PRF.

- $\underline{\mathsf{Hyb}_3}$: In this hybrid, in every execution of the while loop when $\mathsf{guess} = 0$, we make the following changes:
  - For every $i \in H$ and for every $j \in M$, and for each of the m OT executions between P_i and P_j in rounds 1 and 2 of the protocol where P_i acts as the receiver, we generate the receiver OT message from P_i using $\mathsf{Sim}^{eq}_{\mathsf{OT}}$.
  - For every $i \in H$ and for every $j \in H$ and for each of the m OT executions between P_i and P_j in rounds 1 and 2 of the protocol where P_i acts as the receiver, we generate the receiver OT message from P_i as $\mathsf{OT}_1(1^\lambda, 0)$. Similarly, for each the m OT executions where P_i acts as the sender, we compute the second round sender OT message from P_i as $\mathsf{OT}_2(\mathsf{otm}_1, \perp, \perp)$.

  We show in Claim 7.7 that $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$ from the equivocal receiver security and the security in the no corruption setting of the two-round oblivious transfer.

- $\underline{\mathsf{Hyb}_4}$: In this hybrid, in every execution of the while loop when $\mathsf{guess} = 0$, we make the following changes:
  - We non-uniformly fix all the messages in the interaction with $\mathcal{A}$ in the experiment until the adversary sends $\{\pi_1^i\}_{i\in M}$. This fixes the first rounds messages from honest parties in the main thread, the first and second round messages from all the parties in the rewind threads and the first round message from corrupt parties in the main thread. For every $i \in H$ and $j \in M$ and for each of the m OTs between P_i acting as the sender and P_j acting as the receiver in the first two rounds of Π in the main thread, we check if there exists $b \in \{0,1\}$ and $r \in \{0,1\}^*$ such that the first round receiver message from P_j in that OT execution can be written as $\mathsf{OT}_1(1^\lambda, b; r)$.
  - If all the checks pass, then for each $i \in H$ and $j \in M$, we compute the second round sender message from P_i in an execution with P_j as $\mathsf{OT}_2(\mathsf{otm}_1, m_b, m_b)$ where m_b is chosen uniformly. If some check does not pass, then we do not make any changes.

  Relying on the sender security of the oblivious transfer, we show that $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$ in Claim 7.8.

- $\underline{\mathsf{Hyb}_5}$: In this hybrid, we make the following changes:
  - We non-uniformly fix all the messages in the interaction with $\mathcal{A}$ in the experiment up until we send $\{\pi_2^i\}_{i\in H}$. This fixes the first and second round messages from honest parties in the main thread, the first and second round messages from all the parties in the rewind threads and the first round message from corrupt parties in the main thread.
  - If $\chi = 0$ and if $\{x_i, r_i\}_{i\in M}$ sent by $\mathcal{A}$ is consistent with the protocol messages in the main thread, then to generate the fourth round message on behalf of honest P_i, we compute $\mathsf{otm}_{2,i}^{t,\alpha,\beta}$ as $\mathsf{OT}_2(\mathsf{otm}^{t,\alpha,\beta}, \mathsf{lab}^{i,t+1}_{c,r^{t,\alpha,\beta}}, \mathsf{lab}^{i,t+1}_{c,r^{t,\alpha,\beta}})$ for each $t \in [T]$, $\alpha, \beta \in \{0,1\}$ where $\phi_t = (i^*, a, b, c)$.

In Claim 7.9, we show that $\mathsf{Hyb}_5 \approx_c \mathsf{Hyb}_4$ from the sender privacy of the OT protocol.

- $\underline{\mathsf{Hyb}_6}$: In this hybrid, we make the following changes:
  - We non-uniformly fix all the messages in the interaction with $\mathcal{A}$ in the experiment up until we send $\{\pi_2^i\}_{i \in H}$. This fixes the first and round messages from honest parties in the main thread, the first and second round messages from all the parties in the rewind threads and the first round message from corrupt parties in the main thread.
  - If $\chi = 0$, then to generate the third round message on behalf of P_i for each $i \in H$, we compute $\{\mathsf{otm}_1^{t,\alpha,\beta}\}_{t \in A_i, \alpha, \beta in \{0,1\}}$ as the output of $\mathsf{Sim}_{\mathsf{OT}}^{eq}$ and set $\overline{\omega}^{t,\alpha,\beta} = (\omega_0^{t,\alpha,\beta}, \omega_1^{t,\alpha,\beta})$.

  We show that $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$ from the equivocal receiver security property of the oblivious transfer in Claim 7.10.

- $\underline{\mathsf{Hyb}_7}$: In this hybrid, we make the following changes:
  - We non-uniformly fix all the messages in the interaction with $\mathcal{A}$ in the experiment up until we send $\{\pi_2^i\}_{i \in H}$. This fixes the first and round messages from honest parties in the main thread, the first and second round messages from all the parties in the rewind threads and the first round message from corrupt parties in the main thread.
  - If $\{x_i, r_i\}_{i \in M}$ sent by the adversary is consistent with the messages sent in the main thread, then we generate the transcript in the computation phase of Φ where the messages on behalf of the honest parties are generated using their private input and randomness and generate the messages on behalf of the corrupt parties are generated using $\{x_i, r_i\}_{i \in M}$. This generates the transcript Z and the final state st^*. We now make the following changes to the final round message generated on behalf of P_i for each $i \in H$,
    1. For each $k \in [\ell]$, we set $\mathsf{lab}_k^{i,T+1} := 0^\lambda$.
    2. For each t from T down to 1,
       (a) We parse ϕ_t as (i^*, a, b, c).
       (b) We set $\alpha^* := \mathsf{st}_a^*$, $\beta^* := \mathsf{st}_b^*$, and $\gamma^* := \mathsf{st}_c^*$.
       (c) If $i = i^*$ then it parses $\overline{\omega}^{t,\alpha^*,\beta^*}$ as (ω_0, ω_1) and sets $\omega^{t,\alpha^*,\beta^*} = \omega_{\gamma^*}$. It computes
       $$\left(\widetilde{f}^{i,t}, \{\mathsf{lab}_k^{i,t}\}_{k \in [\ell]}\right) \leftarrow \mathsf{Sim}_{\mathsf{Garb}}\left(1^{|f_{i,t}|}, 1^\ell, \left(\gamma^*, \omega^{t,\alpha^*,\beta^*}, \{\mathsf{lab}_k^{i,t+1}\}_{k \in [\ell]}\right)\right).$$
       (d) Else, it computes $\mathsf{otm}_{2,i}^{t,\alpha^*,\beta^*} \leftarrow \mathsf{OT}_2(\mathsf{otm}_1^{t,\alpha^*,\beta^*}, \mathsf{lab}_c^{i,t+1}, \mathsf{lab}_c^{i,t+1})$. It computes
       $$\left(\widetilde{f}^{i,t}, \{\mathsf{lab}_k^{i,t}\}_{k \in [\ell]}\right) \leftarrow \mathsf{Sim}_{\mathsf{Garb}}\left(1^{|f_{i,t}|}, 1^\ell, \left(\mathsf{otm}_{2,i}^{t,\alpha^*,\beta^*}, \{\mathsf{lab}_k^{i,t+1}\}_{k \in [\ell] \setminus \{c\}}\right)\right).$$
    3. It sends $\left\{\left(\{\widetilde{f}^{i,t}\}_{t \in [T]}, \{\mathsf{lab}_k^{i,1}\}_{k \in [\ell]}\right)\right\}_{i \in H}$ to $\mathcal{A}$.

  We show in Claim 7.11 that $\mathsf{Hyb}_7 \approx_c \mathsf{Hyb}_8$ using the security of garbled circuits.

- $\underline{\mathsf{Hyb}_8}$: In this hybrid, we make the following changes:
  - For each $i \in H$, we generate z_i as the output of $\mathsf{Sim}_\phi(i)$.
  - Let $\{x_i, r_i\}_{i \in M}$ be input and randomness of corrupt parties output by $\mathcal{A}$.

- **Case-1: If $\{x_i, r_i\}_{i \in M}$ is consistent.** In this case, on input $\{x_i, r_i\}_{i \in M}$ and $f(x_1, \ldots, x_n)$, we first recover $\{y_i\}_{i \in M}$ and generate the transcript Z and the final state st^* using $\mathsf{Sim}_\Phi(\{y_i\}_{i \in M}, f(x_1, \ldots, x_n))$. We use this output to generate the final round message.
- **Case-2: If $\{x_i, r_i\}_{i \in M}$ is inconsistent.** In this case, we receives the inputs $\{x_i\}_{i \in H}$ and do the following to generate the final round message on behalf of honest P_i
  - We compute the correlations of honest parties as in Hyb_7 and compute the augmented input y_i for each $i \in H$.
  - We compute $v_i \leftarrow \mathsf{Sim}^{eq}_\Phi(z_i, y_i)$.
  - For each $t \in A_i$, $\alpha, \beta \in \{0,1\}$, we compute $r^{t,\alpha,\beta} = (v_{i,c} \oplus \mathsf{NAND}(v_{i,a} \oplus \alpha, v_{i,b} \oplus \beta))$. We parse $\overline{\omega}^{t,\alpha,\beta}$ as (ω_0, ω_1) and set $\omega^{t,\alpha,\beta} = \omega_{r^{t,\alpha,\beta}}$.
  - We generate the round-4 message on behalf of P_i exactly as described in the protocol using $v_i, \{\omega^{t,\alpha,\beta}\}_{t \in A_i, \alpha,\beta \in \{0,1\}}$.

Note that in Case-1, it follows from the perfect security of the conforming protocol that Hyb_7 is identically distributed to Hyb_8. In Case-2, it follows from Property 7.4 that Hyb_7 is identically distributed to Hyb_8.

- Hybrids $\mathsf{Hyb}_9 - \mathsf{Hyb}_{12}$: In these hybrids, we reverse the changes made in Hyb_4-Hyb_1. Note that Hyb_{12} is identically distributed to the output of the ideal experiment.

Claim 7.5. $\mathsf{Hyb}_0 \approx_s \mathsf{Hyb}_1$.

Proof. Note that the only difference between Hyb_0 and Hyb_1 is that Hyb_1 might sometimes output the special symbol fail. We now show that the probability that Hyb_1 outputs the special symbol fail is negligible.

In every execution of the while loop in Hyb_1, the probability that $\chi \neq \mathsf{guess}$ is 1/2. This is because the random variable guess is uniformly distributed given the view of the adversary. Thus, in λ independent iterations, the probability that in every iteration $\chi \neq \mathsf{guess}$ is $2^{-\lambda}$. Thus, Hyb_0 is statistically close to Hyb_1. □

Claim 7.6. *Assuming the security of the PRF, we have* $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$.

Proof. Let $p(\lambda) = (6\lambda)^2$. We first argue that in each execution of the while loop, the probability that $\chi \neq \mathsf{guess}$ in Hyb_2 lies in the interval $[1/2 - 1/p(\lambda), 1/2 + 1/p(\lambda)]$. Assume for the sake of contradiction that the probability that $\chi \neq \mathsf{guess}$ in Hyb_2 does not lie in the above interval. Then, we will give a reduction $\mathcal{B}$ against the security of the PRF. Consider a PRF challenger that for each $i \in H$ and $r \in \{1,2\}$, chooses an independent PRF key k^i_r and when queried on some input x in (i,r)-th instance either gives the output of the PRF applied on x in the real world or gives the output of a random function applied on x in the ideal world. $\mathcal{B}$ does the following:

- It chooses a random bit $\mathsf{guess} \leftarrow \{0,1\}$.
- If $(\mathsf{guess} = 1)$, then for each $i \in H$ and $r \in \{1,2\}$, it chooses an independent PRF key k'^i_r and generates the protocol messages in the first two rounds as in Hyb_1 where the randomness is derived using the PRF.
- If $(\mathsf{guess} = 0)$, then for each $i \in H$ and $r \in \{1,2\}$, $\mathcal{B}$ interacts with the PRF challenger and derives the randomness for generating the protocol messages for P_i in the r-th round by querying the PRF challenger on the transcript seen so far.

- If $(\chi \neq \mathsf{guess})$, then $\mathcal{B}$ outputs 1. Otherwise, it outputs 0.

Note that probability that $\mathcal{B}$ outputs 1 when interacting with an ideal world PRF challenger is the same as the probability that $(\chi \neq \mathsf{guess})$ in Hyb_2. On the other hand, the probability that $\mathcal{B}$ outputs 1 when interacting with the real world PRF challenger is the same as the probability that $(\chi \neq \mathsf{guess})$ in Hyb_1. This probability is 1/2 from Claim 7.5. Thus, if the probability that $\chi \neq \mathsf{guess}$ in Hyb_2 does not lie in the interval $[1/2 - 1/p(\lambda), 1/2 + 1/p(\lambda)]$, then $\mathcal{B}$ breaks the security of the PRF with advantage $1/p(\lambda)$ which is a contradiction.

Thus, in both Hyb_1 and Hyb_2, the probability that $(\mathsf{guess} \neq \chi)$ in each execution of the while loop lies in the interval $[1/2 - 1/p(\lambda), 1/2 + 1/p(\lambda)]$. We will use this fact to show that Hyb_2 is computationally indistinguishable to Hyb_1 from the security of the PRF. Assume for the sake of contradiction that there is a distinguisher D that can distinguish between Hyb_1 and Hyb_2 with non-negligible advantage. Then via a standard averaging argument, there exists two distributions $\mathsf{Hyb}_{1,k}$ and $\mathsf{Hyb}_{1,k-1}$ (described below) such that D can distinguish between $\mathsf{Hyb}_{1,k}$ and $\mathsf{Hyb}_{1,k-1}$ with non-negligible advantage $\mu(\lambda)$. In both $\mathsf{Hyb}_{1,k}$ and $\mathsf{Hyb}_{1,k-1}$, whenever $\mathsf{count} < k$, we generate the messages inside the while loop as in Hyb_2 and when $\mathsf{count} > k$, we generate these messages as in Hyb_1. The only difference is in how the messages corresponding to $\mathsf{count} = k$ are generated. In $\mathsf{Hyb}_{1,k}$, these are generated as in Hyb_2 whereas in $\mathsf{Hyb}_{1,k-1}$, these messages are generated as in Hyb_1. We now use D to give a reduction $\mathcal{C}$ against the security of PRF. $\mathcal{C}$ interacts with a PRF challenger that for each $i \in H$ and $r \in [4]$, chooses an independent PRF key k_r^i and when queried on some input x in (i,r)-th instance either gives the output of $\mathsf{PRF}(k_r^i, x))$ in the real world or gives the output of a random function applied on x in the ideal world. $\mathcal{C}$ does the following in generating the messages in the while loop when $\mathsf{count} = k$:

- It chooses a random bit $\mathsf{guess} \leftarrow \{0,1\}$.
- If $(\mathsf{guess} = 1)$, then for each $i \in H$ and $r \in \{1,2\}$, it chooses an independent PRF key k'^i_r and generates the protocol messages in the first two rounds as in Hyb_1 where the randomness is derived using the PRF.
- If $(\mathsf{guess} = 0)$, then for each $i \in H$ and $r \in \{1,2\}$, $\mathcal{C}$ interacts with an independent PRF challenger and derives the randomness for generating the protocol messages for P_i in the r-th round by querying the PRF challenger on the transcript seen so far.
- If $\chi \neq \mathsf{guess}$, then $\mathcal{C}$ outputs a random bit. If $\chi = \mathsf{guess}$, $\mathcal{C}$ continues with the rest of the execution steps (if $\mathsf{guess} = 0$) where the randomness needed to generate the protocol messages for honest P_i in rounds 3 and 4 are derived by querying the PRF challenger. $\mathcal{C}$ finally runs D on the view of the adversary and the outputs of the honest parties and outputs whatever D outputs.

We now lower bound the probability that $\mathcal{C}$ is able to correctly guess whether it is interacting with the real world or the ideal world PRF challenger.

$$\begin{aligned}
\Pr[\mathcal{C} \text{ correctly guess real/ideal}] &= \Pr[\chi \neq \mathsf{guess}](1/2) + \Pr[\chi = \mathsf{guess}](1/2 + \mu(\lambda)) \\
&= \Pr[\chi \neq \mathsf{guess}](1/2) + (1 - \Pr[\chi \neq \mathsf{guess}])(1/2 + \mu(\lambda)) \\
&= 1/2 + \mu(\lambda) - \Pr[\chi \neq \mathsf{guess}]\mu(\lambda) \\
&> 1/2 + \mu(\lambda) - (1/2 + 1/p(\lambda))\mu(\lambda) \\
&> 1/2 + \mu(\lambda) - (2/3)\mu(\lambda) \\
&= 1/2 + \mu(\lambda)/3
\end{aligned}$$

Thus, $\mathcal{C}$ breaks the security of the PRF with non-negligible advantage $\mu(\lambda)/3$ which is a contradiction. $\square$

Claim 7.7. *Assuming the equivocal receiver security of the oblivious transfer protocol and the security in the no corruption setting, we have* $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$.

Proof. Let $p(\lambda) = (6\lambda)^2$. We first argue that in each execution of the while loop in Hyb_3, the probability that $\chi \neq \mathsf{guess}$ lies in the interval $[1/2 - 2/p(\lambda), 1/2 + 2/p(\lambda)]$. This proof is similar to the one given in Claim 7.6 and relies on a reduction to the equivocal receiver security property of the oblivious transfer and the security in the no corruption setting. In this proof, we consider an intermediate hybrid distribution Hyb'_3 which is same as Hyb_2 except that for every $i \in H$ and for every $j \in M$, and for each of the m OT executions between P_i and P_j in rounds 1 and 2 of the protocol where P_i acts as the receiver, we generate the receiver OT message from P_i using $\mathsf{Sim}^{eq}_{\mathsf{OT}}$. We first show that in Hyb'_3, the probability that $\chi \neq \mathsf{guess}$ lies in the interval $[1/2 - 3/2p(\lambda), 1/2 + 3/2p(\lambda)]$. Suppose this is not the case, then we give a reduction $\mathcal{B}$ that breaks the equivocal receiver security property of the oblivious transfer. $\mathcal{B}$ interacts with a challenger oracle which can be queried multiple times. In each query, $\mathcal{B}$ sends a bit b to the oracle and obtains (otm_1, ω). In the real mode, the response from the oracle is computed using the real algorithms whereas in the ideal mode, the response is computed using $\mathsf{Sim}^{eq}_{\mathsf{OT}}$. The goal of $\mathcal{B}$ is to guess whether it is interacting with the oracle set in the real mode or in the ideal mode. $\mathcal{B}$ interacts with $\mathcal{A}$ and generates the first round messages from honest parties to the corrupt parties by querying the challenger oracle on uniform random bits. If $\chi \neq \mathsf{guess}$, then $\mathcal{B}$ outputs 1 and otherwise, it outputs 0. Note that it follows from Claim 7.6 that in Hyb_2, the probability that $\chi \neq \mathsf{guess}$ lies in the interval $[1/2 - 1/p(\lambda), 1/2 + 1/p(\lambda)]$. This means that $\mathcal{B}$ breaks the equivocal receiver security of the oblivious transfer protocol with advantage $1/2p(\lambda)$ and this is a contradiction. By using a similar argument, we can give a reduction to the security in the no corruption setting and show that in Hyb_3, the probability that $\chi \neq \mathsf{guess}$ in Hyb_3 lies in the interval $[1/2 - 2/p(\lambda), 1/2 + 2/p(\lambda)]$.

Thus, in Hyb_2, Hyb'_3 and Hyb_3, the probability that $\chi \neq \mathsf{guess}$ in each execution of the while loop lies in the interval $[1/2 - 2/p(\lambda), 1/2 + 2/p(\lambda)]$. We now use this fact to show that Hyb_3 is computationally indistinguishable to Hyb_2. To prove this, we show that Hyb_2 is computationally indistinguishable to Hyb'_3 using the equivocal receiver security of the oblivious transfer and that Hyb'_3 is computationally indistinguishable from Hyb_3 from the security in the no corruption setting property.

We start with the former claim. Assume for the sake of contradiction that there exists a distinguisher D that can distinguish between Hyb'_3 and Hyb_2 with non-negligible advantage. As in the proof of Claim 7.6, we can show via an averaging argument that there exists two distributions $\mathsf{Hyb}_{2,k-1}$ and $\mathsf{Hyb}_{2,k}$ such that, (1) these two distributions only differ in how the messages in the k-th execution of the while loop are generated, and (2) D can distinguish between $\mathsf{Hyb}_{2,k-1}$ and $\mathsf{Hyb}_{2,k}$ with non-negligible advantage $\mu(\lambda)$. We now use D to design a reduction $\mathcal{C}$ that break the equivocal receiver security property of the OT protocol. $\mathcal{C}$ is provided access to a challenger oracle which can be queried multiple times. In each query, $\mathcal{C}$ sends a bit b to the oracle and obtains (otm_1, ω). In the real mode, the response from the oracle is computed using the real algorithms whereas in the ideal mode, the response is computed using $\mathsf{Sim}^{eq}_{\mathsf{OT}}$. The goal of $\mathcal{C}$ is to guess whether it is interacting with the oracle set in the real mode or in the ideal mode.

$\mathcal{C}$ does the following in generating the messages in the while loop when $\mathsf{count} = k$:

- It chooses a random bit $\mathsf{guess} \leftarrow \{0, 1\}$.
- If $(\mathsf{guess} = 1)$, then for each $i \in H$ and $r \in \{1, 2\}$, it generates the first two round protocol messages as in $\mathsf{Hyb}_{2,k-1}$.

- If $(\mathsf{guess} = 0)$, then for each $i \in H$, for each $j \in M$ and for each of the m OT executions, $\mathcal{C}$ chooses a random bit b and sends this to the oracle. $\mathcal{C}$ receives (otm_1, ω). It uses otm_1 to generate the first round message on behalf of honest P_i.

- If $\chi \neq \mathsf{guess}$, then $\mathcal{C}$ outputs a random bit. If $\chi = \mathsf{guess}$, $\mathcal{C}$ continues with the rest of the execution steps (if $\mathsf{guess} = 0$) and uses ω received from the challenger to compute the receiver OT correlation at the end of the second round. $\mathcal{C}$ finally runs D on the view of the adversary and the outputs of the honest parties and outputs whatever D outputs.

Since $2/p(\lambda) < 1/6$, we can use the exact same analysis as in the proof of claim 7.6 and argue that $\mathcal{C}$ breaks the equivocal receiver security with advantage $\mu(\lambda)/3$.

To show that Hyb_3' is computationally indistinguishable to Hyb_3, the argument is similar to the one above. The only difference is that when $\mathsf{guess} = 0$, we provide the external challenger oracle with two sets of inputs $(0, \perp, \perp)$ and (b, m_0, m_1) where b, m_0, m_1 are uniformly chosen. The oracle provides with $\mathsf{otm}_1, \mathsf{otm}_2$ and then we use this to generate the messages in the first two rounds. When $\mathsf{guess} = \chi$, to generate the round-3 and round-4 messages on behalf of the honest parties, we use (b, m_b) as the receiver correlation and (m_0, m_1) as the sender correlation. □

Claim 7.8. *Assuming the sender security property of the OT protocol, we have* $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$.

Proof. Let $p(\lambda) = (6\lambda)^2$. Using a similar argument given in Claim 7.6, we can rely on the sender security of the OT protocol and show that in each execution of the while loop, the probability that $\chi \neq \mathsf{guess}$ in Hyb_4 lies in the interval $[1/2 - 3/p(\lambda), 1/2 + 3/p(\lambda)]$.

Thus, in both Hyb_3 and in Hyb_4, the probability that $(\chi \neq \mathsf{guess})$ in each invocation of the while loop lies in the interval $[1/2 - 3/p(\lambda), 1/2 + 3/p(\lambda)]$. Assume for the sake of contradiction that there exists a PPT distinguisher D that can distinguish between Hyb_3 and Hyb_4 with non-negligible advantage. As in the proof of Claim 7.6, we can show via an averaging argument that there exists two distributions $\mathsf{Hyb}_{2,k-1}$ and $\mathsf{Hyb}_{2,k}$ such that, (1) these two distributions only differ in how the messages in the k-th execution of the while loop are generated, and (2) D can distinguish between $\mathsf{Hyb}_{2,k-1}$ and $\mathsf{Hyb}_{2,k}$ with non-negligible advantage $\mu(\lambda)$. We now use D to design an adversary $\mathcal{C}$ that breaks the sender security of the special OT protocol. $\mathcal{C}$ is provided access to a challenger oracle which can be queried multiple times. In each query, $\mathcal{C}$ sends $\mathsf{otm}_1, (b, r), (m_0, m_1))$ to the oracle (where $\mathsf{otm}_1 := \mathsf{OT}_1(1^\lambda, b; r)$) and obtains otm_2. In the real mode, the response from the oracle is computed as $\mathsf{OT}_1(\mathsf{otm}_1, m_0, m_1)$ whereas in the ideal mode, the response is computed as $\mathsf{OT}_2(\mathsf{otm}_1, m_b, m_b)$. The goal of $\mathcal{C}$ is to guess whether it is interacting with the oracle set in the real mode or in the ideal mode.

$\mathcal{C}$ does the following in generating the messages in the while loop when $\mathsf{count} = k$:

- It chooses a random bit $\mathsf{guess} \leftarrow \{0, 1\}$.

- If $(\mathsf{guess} = 1)$, then for each $i \in H$ and $r \in \{1, 2\}$, it generates the first two round protocol messages as in $\mathsf{Hyb}_{3,k-1}$.

- If $(\mathsf{guess} = 0)$, we non-uniformly fix the messages until $\mathcal{A}$ sends $\{\pi_1^i\}_{i \in [m]}$. For every $i \in H$ and $j \in M$ and for each of the m OTs between P_i acting as the sender and P_j acting as the receiver in the first two rounds of Π in the main thread, we check if there exists $b \in \{0, 1\}$ and $r \in \{0, 1\}^*$ such that the first round receiver message from P_j in that OT execution can be written as $\mathsf{OT}_1(1^\lambda, b; r)$. If some checks do not pass, then $\mathcal{C}$ generates the second round OT message as in $\mathsf{Hyb}_{3,k-1}$.

- For each of the received first round OT message from a corrupt party, $\mathcal{C}$ queries the challenger oracle on this message, b, r (obtained non-uniformly) and two independently chosen random inputs m_0, m_1. It receives otm_2 from the challenger and uses this to generate the second round message of the protocol.

- If $\chi \neq \mathsf{guess}$, then $\mathcal{C}$ outputs a random bit. If $\chi = \mathsf{guess}$, $\mathcal{C}$ continues with the rest of the execution steps as in $\mathsf{Hyb}_{3,k-1}$. $\mathcal{C}$ finally runs D on the view of the adversary and the outputs of the honest parties and outputs whatever D outputs.

Since $3/p(\lambda) < 1/6$, we can use the exact same analysis as in the proof of claim 7.6 and argue that $\mathcal{C}$ breaks the sender security with advantage $\mu(\lambda)/3$. $\square$

Claim 7.9. *Assuming the sender privacy property of the oblivious transfer, we have* $\mathsf{Hyb}_4 \approx_s \mathsf{Hyb}_5$.

Proof. Assume for the sake of contradiction that there exists a distinguisher D that can distinguish between Hyb_4 and Hyb_5 with non-negligible advantage. We now give an adversary $\mathcal{C}$ that uses D and breaks the sender privacy property of the OT. $\mathcal{C}$ is provided access to a challenger oracle which can be queried multiple times. In each query, $\mathcal{C}$ sends $\mathsf{otm}_1, (b, r), (m_0, m_1))$ to the oracle and obtains otm_2. In the real mode, the response from the oracle is computed as $\mathsf{OT}_1(\mathsf{otm}_1, m_0, m_1)$ whereas in the ideal mode, the response is computed as $\mathsf{OT}_2(\mathsf{otm}_1, m_b, m_b)$ where $b \in \{0,1\}$ and $r \in \{0,1\}^*$ are such that $(\mathsf{otm}_1, \cdot) \leftarrow \mathsf{OT}_2(1^\lambda, b; r)$. The goal of $\mathcal{C}$ is to guess whether it is interacting with the oracle set in the real mode or in the ideal mode.

$\mathcal{C}$ interacts with $\mathcal{A}$ as in the Hyb_4 until it has to send the last round message on behalf the honest parties. If $\chi \neq 0$ or if $\{x_i, r_i\}_{i \in M}$ provided by adversary are not consistent with the messages in the main thread, then Hyb_4 and Hyb_5 are identically distributed. Let us assume that this is not the case. For each $t \in [T]$, $\alpha, \beta \in \{0,1\}$, and for each $i \in H$, $\mathcal{C}$ sends a query to the oracle with the input $(\mathsf{otm}_1^{t,\alpha,\beta}, (r^{t,\alpha,\beta}, s^{t,\alpha,\beta}), \mathsf{lab}_{c,0}^{i,t+1}, \mathsf{lab}_{c,1}^{i,t+1})$ (where $\phi_t = (i^*, a, b, c)$, $r^{t,\alpha,\beta}, s^{t,\alpha,\beta}$ (obtained from $\{x_{i^*}, r_{i^*}\}$ if $i^* \in M$ or obtained from the input and random tape of i^* if $i^* \in H$) are such that $\mathsf{otm}_1^{t,\alpha,\beta} := \mathsf{OT}_1(1^\lambda, r^{t,\alpha,\beta}; s^{t,\alpha,\beta})$). The oracle gives $\mathsf{otm}_{2,i}^{t,\alpha,\beta}$ as the response. $\mathcal{C}$ uses this to generate the final round message on behalf of each honest P_i. $\mathcal{C}$ finally runs D on the view of the adversary and the outputs of the honest parties and outputs whatever D outputs.

Note that if the oracle generated the responses in the ideal mode, then the inputs to D are distributed identically to Hyb_5. Otherwise, they are identically distributed to Hyb_4. Thus, if D can distinguish between Hyb_4 and Hyb_5 with non-negligible advantage, then $\mathcal{C}$ can break the sender privacy of the OT with the same advantage which is a contradiction. $\square$

Claim 7.10. *Assuming the equivocal receiver security of the oblivious transfer protocol, we have* $\mathsf{Hyb}_5 \approx_c \mathsf{Hyb}_6$.

Proof. Assume for the sake of contradiction that there exists a distinguisher D that can distinguish between Hyb_5 and Hyb_6 with non-negligible advantage. We now give an adversary $\mathcal{C}$ that uses D and breaks the equivocal receiver security of the oblivious transfer. $\mathcal{C}$ is provided access to a challenger oracle which can be queried multiple times. In each query, $\mathcal{C}$ sends a bit b to the oracle and obtains (otm_1, ω). In the real mode, the response from the oracle is computed using the real algorithms whereas in the ideal mode, the response is computed using $\mathsf{Sim}_{\mathsf{OT}}^{eq}$. The goal of $\mathcal{C}$ is to guess whether it is interacting with the oracle set in the real mode or in the ideal mode.

If $\chi = 0$, we now explain how $\mathcal{C}$ generates the messages on behalf of the honest parties. For each $i \in H$, $t \in A_i$ and $\alpha, \beta \in \{0,1\}$, $\mathcal{C}$ makes a query to the challenge oracle with the input bit $r^{t,\alpha,\beta}$ and receives $(\mathsf{otm}_1^{r^{t,\alpha,\beta}}, \omega^{t,\alpha,\beta})$. $\mathcal{C}$ uses $\mathsf{otm}_1^{r^{t,\alpha,\beta}}$ to generate the third round message and uses

$\omega^{t,\alpha,\beta}$ to generate the final round message. $\mathcal{C}$ finally runs D on the view of the adversary and the outputs of the honest parties and outputs whatever D outputs.

Note that if the oracle generated the responses in the ideal mode, then the inputs to D are distributed identically to Hyb_6. Otherwise, they are identically distributed to Hyb_5. Thus, if D can distinguish between Hyb_5 and Hyb_6 with non-negligible advantage, then $\mathcal{C}$ can break the equivocal receiver security property of the OT with the same advantage which is a contradiction. □

Claim 7.11. *Assuming the security of garbled circuits, we have* $\mathsf{Hyb}_6 \approx_c \mathsf{Hyb}_7$.

Proof. Assume for the sake of contradiction that there exists a distinguisher D that can distinguish between the outputs of Hyb_6 and Hyb_7 with non-negligible advantage. Then, by a standard averaging argument there exists two distributions $\mathsf{Hyb}_{6,t-1}$ and $\mathsf{Hyb}_{6,t}$ for some $t \in [T]$ such that $\mathsf{Hyb}_{6,0} \equiv \mathsf{Hyb}_6$ and $\mathsf{Hyb}_{6,T} \equiv \mathsf{Hyb}_7$ such that (1) they differ only in how the garbled circuits generated corresponding to round-t of the conforming protocol is generated. Specifically, in $\mathsf{Hyb}_{6,t}$ they are simulated and in $\mathsf{Hyb}_{6,t-1}$ they are generated honestly. For every $t^* > t$, the garbled circuits in both these distributions are generated honestly and for every $t^* < t$, the garbled circuits in both these distributions are generated using the simulator; (2) D can distinguish between $\mathsf{Hyb}_{6,t-1}$ and $\mathsf{Hyb}_{6,t}$ with non-negligible advantage. We now use D to construct an adversary $\mathcal{C}$ that breaks the security of garbled circuits. $\mathcal{C}$ is given access to an oracle that can be queried multiple times. When $\mathcal{C}$ sends a query with a circuit C and an input x, the oracle returns $\widetilde{C}, \{\mathsf{lab}_k\}_{k\in|x|}$. In the real mode, the query is answered using the actual garbling algorithm and in the ideal mode, these are generated using the simulator for the garbling scheme. The goal of $\mathcal{C}$ is to guess whether the oracle is in the real mode or in the ideal mode.

$\mathcal{C}$ interacts with $\mathcal{A}$ as in the Hyb_6 until it has to send the last round message on behalf the honest parties. If $\chi \neq 0$ or if $\{x_i, r_i\}_{i\in M}$ provided by adversary are not consistent with the messages in the main thread, then Hyb_6 and Hyb_7 are identically distributed. Let us assume that this is not the case. $\mathcal{C}$ computes the transcript Z and the final state st^* as described in the protocol. To generate the garbled circuits corresponding to round t of the computation phase, $\mathcal{C}$ does the following:

- For each $t^* > t$, it generates the garbled circuits for round t^* as in $\mathsf{Hyb}_{6,t-1}$.
- Let $\phi_t := (i^*, a, b, c)$.
- We set $\alpha^* := \mathsf{st}^*_a$, $\beta^* := \mathsf{st}^*_b$, and $\gamma^* := \mathsf{st}^*_c$.
- Let st^*_t be the state of the parties at the beginning of the t-th round.
- If $i^* \in H$, then $\mathcal{C}$ parses $\overline{\omega}^{t,\alpha,\beta}$ as (ω_0, ω_1) for each $\alpha, \beta \in \{0,1\}$ and sets $\omega^{t,\alpha^*,\beta^*} = \omega_{\gamma^*}$ sends a query to its oracle with the input $f^{i^*,t}[v_{i^*}, \{\omega^{t,\alpha,\beta}\}_{\alpha,\beta}, \perp, \mathsf{lab}^{i^*,t+1}]$ and st^*_t. It obtains $(\widetilde{f}^{i^*,t}, \{\mathsf{lab}^{i^*,t}_k\}_{k\in[\ell]})$.
- For every $i \in H \setminus \{i^*\}$, $\mathcal{C}$ queries the oracle on input $f^{i,t}[v_i, \perp, \{\mathsf{otm}^{t,\alpha,\beta}_{2,i}\}_{\alpha,\beta}, \mathsf{lab}^{i,t+1}]$ and $\mathsf{st}^*_t s$ and obtains $(\widetilde{f}^{i,t}, \{\mathsf{lab}^{i,t}_k\}_{k\in[\ell]})$.
- For each $t^* < t$, it generates the garbled circuit for round t^* as in $\mathsf{Hyb}_{6,t-1}$.
- $\mathcal{C}$ finally runs the distinguisher D on the view of the adversary and the outputs of the honest parties and outputs whatever D outputs.

Note that if the oracle generated the responses in the ideal mode, then the inputs to D are distributed identically to Hyb_7. Otherwise, they are identically distributed to Hyb_6. Thus, if D can distinguish between Hyb_7 and Hyb_6 with non-negligible advantage, then $\mathcal{C}$ can break the security of garbled circuits with the same advantage which is a contradiction. □

8 Two-Round Protocol in the Watchlist Correlations Model

In this section, we describe a simple two-round protocol in watchlist correlations model that securely computes arbitrary multiparty functionalities. This protocol makes black-box use of a two-round, semi-malicious MPC protocol with first message equivocality (see the formal definition below).

We start with the description of the building blocks.

- We use a 2-round, n-client, m-server MPC protocol satisfying privacy with knowledge of output property (see Remark A.3) against a malicious, adaptive adversary corrupting up to $n-1$ clients and $t = (m-1)/3$ servers. Such a protocol was constructed in [IKP10, Pas12] making black-box use of a pseudorandom generator (PRG). We set $m = 16\lambda n^2$. See Section 6.1.1 for the syntax of this protocol.

- We use a two-round inner protocol $(\Pi_1, \Pi_2, \mathsf{out}_\Pi)$ with the following syntax. Π_1 takes the index i for the party, its private input x_i, uses random tape r_i and outputs the first round message π_1^i along with an output key sk_i. Π_2 takes the transcript of the first round $\pi(1)$, the private input x_i, uses random tape r_i and outputs the second round message π_2^i. The output function out_Π takes the index i, the transcript $\pi(2)$ of the first two rounds and the output key sk_i and gives the output of the functionaly.

  Definition 8.1. *We say that* $(\Pi_1, \Pi_2, \mathsf{out}_\Pi)$ *is a two-round, inner protocol for computing a function* f *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* $i \in [n]$*,*

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

    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 10 and for each $i \in H$*,* r_i *is uniformly chosen.*

Theorem 8.2. *Let* f *be an arbitrary multiparty functionality and assume black-box access to a two-round inner protocol for computing arbitrary multiparty functions satisfying Definition 8.1. Then, there exists a two-round protocol that computes* f *against static, malicious adversaries satisfying security with selective abort in the watchlist correlations model (described in Figure 11). 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 where communication is over point-to-point channels.*

For simplicity of exposition, we give the description of the protocol over a broadcast channel. As in the case of our five-round protocol, we can relax the communication channel to be point-to-point by relying on the special CDS protocol given in Section F.1. We also note that the watchlist correlation functionality is universal and is “short." Specifically, the size of the watchlist correlations setup phase in our protocol is dependent only on the security parameter and the number of parties and is otherwise, independent of the size of the function to be computed.

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

1. For each $i \in H$, compute $(\pi_1^i, sk_i) := \Pi_r(1^\lambda, i, x_i; r_i)$.
2. Send $\{\pi_1^i\}_{i \in H}$ to $\mathcal{A}$.
3. Receive $\{\pi_1^i, (x_i, r_i)\}_{i \in M}$ from $\mathcal{A}$.
4. Check if the messages sent by corrupt parties in $\pi(1)$ are consistent with $\{x_i, r_i\}_{i \in M}$.
5. **Semi-Malicious Security:** If they are consistent:
   (a) For each $i \in H$, compute $\pi_2^i := \Pi_2(1^\lambda, i, x_i, \pi(1); r_i)$.
6. **Equivocality:** If they are not consistent:
   (a) For each $i \in H$, compute $\pi_2^i := \Pi_2(1^\lambda, i, x_i, \pi(1); r_i)$.
7. Send $\{\pi_2^i\}_{i \in H}$ to $\mathcal{A}$.
8. Receive $\{\pi_2^i\}_{i \in M}$ from $\mathcal{A}$.
9. Output the view of $\mathcal{A}$ and $\{\mathsf{out}_\Pi(i, \pi(2), sk_i)\}_{i \in H}$.

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

1. For each $i \in H$, compute $\pi_1^i := \mathsf{Sim}_\Pi(1^\lambda, i)$.
2. Send $\{\pi_1^i\}_{i \in H}$ to $\mathcal{A}$.
3. Receive $\{\pi_1^i, (x_i, r_i)\}_{i \in M}$ from $\mathcal{A}$.
4. Check if the messages sent by corrupt parties in $\pi(1)$ are consistent with $\{x_i, r_i\}_{i \in M}$.
5. **Semi-Malicious Security:** If they are consistent:
   (a) For each $i \in H$, compute $\pi_2^i \leftarrow \mathsf{Sim}_\Pi(1^\lambda, i, f(x_1, \ldots, x_n), \{x_j, r_j\}_{j \in M}, \pi(1))$.
6. **Equivocality:** If they are not consistent:
   (a) For each $i \in H$, compute $\pi_2^i \leftarrow \mathsf{Sim}_\Pi(1^\lambda, i, \{x_i\}_{i \in H}, \pi(1))$.
7. Send $\{\pi_2^i\}_{i \in H}$ to $\mathcal{A}$.
8. Receive $\{\pi_2^i\}_{i \in M}$ from $\mathcal{A}$.
9. If the transcript $\pi(2)$ is consistent with $\{x_i, r_i\}_{i \in M}$, output view of $\mathcal{A}$ and $\{f(x_1, \ldots, x_n)\}_{i \in H}$. Else, run $\mathsf{Sim}_\Pi(\{x_i\}_{i \in H}, \pi(2))$ to obtain $\{sk_i\}_{i \in H}$ and output view of view of $\mathcal{A}$ and $\{\mathsf{out}_\Pi(i, \pi(2), sk_i)\}_{i \in H}$.

Figure 10: Security Game for the Two-Round Inner Protocol

8.1 Construction

We now describe our two-round, black-box MPC construction in Figure 12 in the watchlist correlations model. Let n be the number of parties and let f be the multiparty function to be securely computed. Let f' be a related functionality that takes $z_i := (\chi_i, k_i)$ from P_i where k_i is a MAC key and computes $y = f(\chi_1, \ldots, \chi_n)$ and outputs $(y, \mathsf{MAC}(k_1, y), \ldots, \mathsf{MAC}(k_n, y))$. The protocol for computing f makes use of the outer MPC protocol $(\Phi_1, \Phi_2, \mathsf{out}_\Phi)$ for securely computing the function f', m instances of the inner MPC protocol where i-th instance $\Pi_i = (\Pi_{i,1}, \Pi_{i,2}, \mathsf{out}_{\Pi_i})$ implements the computation done by the i-th server in the outer protocol.

8.2 Simulator

In this subsection, we give the description of the ideal world simulator Sim for our black-box MPC protocol.

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

- **Watchlist Correlation Setup:**
  - For every $i \in M$ and $j \in H$, Sim intercepts the query $K_{i,j}$ that corrupt P_i sends as the receiver to the watchlist correlation functionality where P_j acts as the sender. Sim samples $\{k_h^{i,j}\}_{h \in K_{i,j}}$ uniformly at random and sends this to P_i.
  - For every $i \in H$ and $j \in M$, Sim intercepts the query $\{k_1^{i,j}, \ldots, k_m^{i,j}\}$ that P_j sends to the watchlist correlations functionality and records it.

The watchlist correlations model is defined by the following ideal functionality for generating a multiparty correlation between a receiver P_i and $n-1$ senders $P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_n$. The setup for the watchlist correlations model includes n independent instances of this functionality, where in each instance a different party P_i acts as a receiver.
Let $\mathcal{A}$ be an adversary that corrupts a set $M \subset [n]$ of the parties and let $H = [n] \setminus M$ be the set of uncorrupted parties. The functionality is parameterized by $m = m(\lambda)$ and $\kappa = \kappa(\lambda)$

- If $P_i \in H$, then for every $j \in [n] \setminus \{i\}$ s.t. $j \in M$, P_i receives m strings $(k_1^{i,j}, \ldots, k_m^{i,j})$ from $\mathcal{A}$.
- For each $j \in [n] \setminus \{i\}$ s.t. $j \in H$, the functionality samples uniform random strings $(k_1^{i,j}, \ldots, k_m^{i,j})$.
- If $P_i \in H$, then the functionality does the following. It samples a set $K_i \subseteq [m]$ of size κ uniformly. It outputs $(K_i, \{k_h^{i,j}\}_{j\in[n]\setminus\{i\}, h\in K_i})$ to P_i and $(k_1^{i,j}, \ldots, k_m^{i,j})$ to P_j for each $j \in [n] \setminus \{i\}$.
- If $P_i \in M$, then the functionality receives from $\mathcal{A}$ a set $K_{i,j}$ of size κ for each $j \in [n] \setminus \{i\}$. It outputs $(K_{i,j}, \{k_h^{i,j}\}_{j\in([n]\setminus\{i\})\cap H, h\in K_{i,j}})$ to P_i and $(k_1^{i,j}, \ldots, k_m^{i,j})$ to P_j for each $j \in ([n] \setminus \{i\}) \cap H$.

Figure 11: Watchlist Correlations Model

- 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 $\{\phi_1^{i\to j}\}_{i\in H, j\in C}$.
- For each $h \in C$, Sim chooses an uniform random tape $\{r_{i,h}\}_{i\in H}$. For every $i \in H$ and $h \in C$, Sim uses $\phi_1^{i\to h}$ as the input and $r_{i,h}$ as the random tape of P_i to generate the first round message in the protocol Π_h.
- For each $i \in H$ and $j \in M$, Sim computes $\mathsf{ct}_h^{j,i}$ as per the protocol if $h \in K_{j,i}$ using the key $k_h^{j,i}$. Otherwise, it generates it as an encryption of a junk value under a uniformly chosen key. For each $i \in H$ and $j \in H$, Sim computes $\mathsf{ct}_h^{j,i}$ as an encryption of a junk value for each $h \in [m]$ under a uniformly chosen key.
- 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 $\mathcal{A}$.
- It uses the sampled keys $\{k_h^{j,i}\}_{j\in H, i\in M, h\in[m]}$ in the watchlist correlations setup phase to compute $\{y_{i,j}\}_{i\in M, j\in H}$ from the ciphertexts $\{ct_h^{j,i}\}_{j\in H, i\in M, h\in[m]}$.
- For each $h \in [m]$, Sim checks if there exists some $j \in H$ such that for every $i \in M$, $y_{i,j}$ 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 $(\phi_1^{i\to h}, r_{i,h})$ present in $y_{i,j}$ as the consistent input and randomness used by corrupt party P_i in the protocol Π_h.

- **Watchlist Correlations Setup:** For each $i \in [n]$, P_i does the following:
  1. P_i invokes the watchlist correlation functionality with parameter m and $\kappa = 2\lambda$ acting as the receiver and for every $j \in [n] \setminus \{i\}$, P_j acting as the sender.
  2. P_i receives $(K_i, \{k_h^{i,j}\}_{j\in[n]\setminus\{i\},h\in K_i})$ from the functionality and for every $j \in [n] \setminus \{i\}$, P_j obtains $(k_1^{i,j}, \ldots, k_m^{i,j})$ from the functionality.
- **Round-1:** In the first round, the party P_i with input χ_i does the following:
  1. It chooses a random MAC key $k_i \leftarrow \{0,1\}^*$ and sets $z_i := (\chi_i, k_i)$.
  2. It computes $(\phi_1^{i\to1}, \ldots, \phi_1^{i\to m}) \leftarrow \Phi_1(1^\lambda, i, z_i)$.
  3. It chooses a random string $r_{i,h} \leftarrow \{0,1\}^*$ for every $h \in [m]$ and sets $y_{i,j} = \{r_{i,h}, \phi_1^{i\to h}\}_{h\in[m]}$ for every $j \in [n] \setminus \{i\}$.
  4. For each $j \in [n] \setminus \{i\}$ and $h \in [m]$, it computes $\mathsf{ct}_h^{j,i} := \mathsf{Enc}_{k_h^{j,i}}(r_{i,h}, \phi_1^{i\to h})$.
  5. For each $h \in [m]$, it computes $(\pi_{h,1}^i, sk_h^i) := \Pi_{h,1}(1^\lambda, i, \phi_1^{i\to h}; r_{i,h})$.
  6. It broadcasts $\{\pi_{h,1}^i, \{\mathsf{ct}_h^{j,i}\}_{j\in[n]\setminus\{i\}}\}_{h\in[m]}$.
- **Round-2:** In the second round, P_i does the following:
  1. It decrypts $\{\mathsf{ct}_h^{i,j}\}_{j\in[n]\setminus\{i\},h\in K_i}$ using $\{k_h^{i,j}\}_{j\in[n]\setminus\{i\},h\in K_i}$ to obtain $\{r_{j,h}, \phi_1^{j\to h}\}_{j\in[n]\setminus\{i\},h\in K_i}$.
  2. Choose a random subset K_i' of K_i having size λ.
  3. For each $j \in [n] \setminus \{i\}$ and $h \in K_i'$, it checks:
     (a) If the PRG computations in $\phi_1^{j\to h}$ are correct.
     (b) If $\pi_{h,1}^j := \Pi_{h,1}(1^\lambda, j, \phi_1^{j\to h}; r_{j,h})$.
  4. If any of the above checks fail, it aborts.
  5. Else, for each $h \in [m]$, it computes $\pi_{h,2}^i := \Pi_{h,2}(1^\lambda, i, \phi_1^{i\to h}, \pi_h(1); r_{i,h})$.
  6. 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. For every $j \in [n] \setminus \{i\}$ and $h \in K_i \setminus K_i'$,
     (a) If the PRG computations in $\phi_1^{j\to h}$ are correct.
     (b) For each $\ell \in [2]$, whether $\pi_{h,\ell}^j := \Pi_{h,\ell}(1^\lambda, j, \phi_1^{j\to h}, \pi_h(\ell-1); r_{j,h})$ where $\pi_h(0)$ is set to be the null string.
  3. If any of the checks fail, then abort.
  4. Else, for every $h \in [m]$, it computes $\phi_2^h := \mathsf{out}_{\Pi_h}(i, \pi_h(2), sk_h^i)$.
  5. It computes $\mathsf{out}_\Phi(\{\phi_2^h\}_{h\in[m]})$ to recover $(y, \sigma_1, \ldots, \sigma_n)$.
  6. 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 12: Description of the Two-Round MPC Protocol in the Watchlist Correlations Model

- If $|C'| > \lambda n^2$, then Sim instructs the ideal functionality to send abort to all the honest parties and outputs the view of the adversary.
- If $|C'| \leq \lambda n^2$, then Sim instructs Sim_Φ to adaptively corrupt the servers indexed by C' and obtains $\{\phi_1^{i\to h}\}_{i\in H,h\in C'}$. For each $i \in H$, Sim chooses a random subset K_i of $[m]$ of size 2λ and a random subset $K_i' \subseteq K_i$ of size λ. If for any $h \in K_i'$, $\{y_{j,i}\}_{j\in M}$ contains inconsistent input and randomness, then Sim instructs the ideal functionality to send abort to i. Let H'

be the subset of honest parties that have not aborted.

- For every $h \in [m] \setminus \{C \cup C'\}$, Sim sends $\{\phi_1^{i \to h}\}_{i \in M}$ to Sim_Φ. Sim_Φ queries the ideal functionality f' on inputs $\{z_i := (\chi_i, k_i)\}_{i \in M}$. Sim sends $\{\chi_i\}_{i \in M}$ to its ideal functionality f and obtains y. 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 $\{\phi_2^h\}_{h \in [m] \setminus \{C \cup C'\}}$.

- To generate the final round message,
  - For each $h \in [m] \setminus \{C \cup C'\}$, Sim sends $\{(\phi_1^{i \to h}, r_{i,h})\}_{i \in M}$ as the input and the randomness of corrupt parties and ϕ_2^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 $\{\phi_1^{i \to h}\}_{i \in H}$ 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 $\{r_{i,h}, \phi_1^{i \to h}\}_{i \in H}$ to generate the final round message on behalf of $H' \subseteq H$.

- 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 $h \in [m]$, Sim checks if there exists some $j \in H$ such that for every $i \in M$, $y_{i,j}$ 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 $|C''| > \lambda n^2$, then Sim instructs the ideal functionality to send abort to all the honest parties and outputs the view of the adversary.
  - Else, if $|C''| \leq \lambda n^2$, then Sim instructs Sim_Φ to adaptively corrupt the servers indexed by C'' and obtains $\{\phi_1^{i \to h}\}_{i \in H, h \in C''}$. If for any $h \in K_i \setminus K'_i$, $\{y_{j,i}\}_{j \in M}$ contains inconsistent input and randomness, then Sim instructs the ideal functionality to send abort to i.
  - For each $h \in C' \cup C''$, Sim runs Sim_{Π_h} on inputs $\{\Phi_1^{i \to h}\}_{i \in H}$ to obtain sk_h^i.
  - For each $h \in C \cup C' \cup C''$, Sim computes ϕ_2^h as $\mathsf{out}_{\Pi_h}(i, \pi_h(2), sk_h^i)$.
  - It then computes $(y', \sigma'_1, \ldots, \sigma'_n) := \mathsf{out}_\Phi(\{\phi_2^h\}_{h \in [m]})$.
  - 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.

8.3 Proof of Indistinguishability

We now show that the real and ideal worlds are computationally indistinguishable via 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 $h \notin K_{j,i}$, we generate $\mathsf{ct}_h^{j,i}$ as an encryption of a junk value under a uniformly chosen key.
- If $j \in H$, then for each $h \in [m]$, we generate $\mathsf{ct}_h^{j,i}$ as an encryption of a junk value under a uniformly chosen key.

The computational indistinguishability between Hyb_0 and Hyb_1 follows immediately from the semantic security of the encryption scheme.

- $\underline{\mathsf{Hyb}_2}$: 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 sampled keys $\{k_h^{j,i}\}_{j \in H, i \in M, h \in [m]}$ in the watchlist correlations setup phase to compute $\{y_{i,j}\}_{i \in M, j \in H}$ from the ciphertexts $\{ct_h^{j,i}\}_{j \in H, i \in M, h \in [m]}$.
  - For each $h \in [m]$, we check if there exists some $j \in H$ such that for every $i \in M$, $y_{i,j}$ 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 $(\phi_1^{i \to h}, r_{i,h})$ present in $y_{i,j}$ as the consistent input and randomness used by corrupt party P_i in the protocol Π_h.
  - For each $i \in H$, we chooses a random subset K_i of $[m]$ of size 2λ and a random subset $K_i' \subseteq K_i$ of size λ. If for any $h \in K_i'$, $\{y_{j,i}\}_{j \in M}$ contains inconsistent input and randomness, 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 $\{(\phi_1^{i \to h}, r_{i,h})\}_{i \in M}$ as the input and the randomness of corrupt parties and ϕ_2^h (computed honestly using $\{\phi_1^{i \to h}\}_{i \in H}$) 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 $\{\phi_1^{i \to h}\}_{i \in H}$ 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 $\{r_{i,h}, \phi_1^{i \to h}\}_{i \in H}$ 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 simulation.

  We show in Claim 8.3 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' and C'' as in the simulation and if $|C'| > \lambda n^2$ or if $|C''| > \lambda n^2$, then we abort.

  Note that the set C' and C'' are random subsets of $[m]$ of size λ. It follows via an identical proof to Claim 6.7 that if either $|C'| > \lambda n^2$ or if $|C''| > \lambda n^2$, then every honest party aborts with overwhelming probability and thus, $\mathsf{Hyb}_2 \approx_s \mathsf{Hyb}_3$.

- $\underline{\mathsf{Hyb}_4}$: 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 8.4 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 f' on $\{\chi_i, k_i\}_{i \in M}$, we query f on $\{\chi_i\}_{i \in M}$ 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.

Note that Hyb_5 is identically distributed to the ideal execution. We can show via an identical argument to the one given in Claim 6.9 that Hyb_4 is statistically close to Hyb_5 from the security of the MAC scheme.

Claim 8.3. *Assuming the security of the inner MPC protocol, we have* $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$*.*

Proof. 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 $\{\phi_1^{i \to h}\}_{i \in H}$ as the inputs of the honest parties. It obtains the first round message $\{\pi^i_{h,1}\}$ 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 watchlist correlations setup phase to compute $\{y_{i,j}\}_{i \in M, j \in H}$ from the ciphertexts $\{ct_h^{j,i}\}_{j \in H, i \in M, h \in [m]}$ received from $\mathcal{A}$.

$\mathcal{B}$ checks if there exists some $j \in H$ such that for every $i \in M$, $y_{i,j}$ 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 $(\phi_1^{i \to h}, r_{i,h})$ present in $y_{i,j}$ 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 $\{\pi^i_{h,2}\}_{i \in H}$ 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, it checks if the input and randomness $\{(\phi_1^{i \to h}, r_{i,h})\}_{i \in M}$ is consistent with the second round messages $\{\pi^i_{h,2}\}_{i \in M}$ received from $\mathcal{A}$. If not, it sends $\{\phi_1^{i \to h}\}_{i \in H}$ to the challenger. It finally obtains the output ϕ_2^h. It computes $\{\phi_2^{h'}\}_{h' \neq h}$ as in $\mathsf{Hyb}_{1,h-1}$ and uses this to compute the output of each honest party. 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. □

Claim 8.4. *Assuming the privacy with knowledge of outputs property of the outer MPC protocol, we have* $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$.

Proof. 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.

$\mathcal{B}$ chooses a random MAC key k_i for each $i \in H$ and sends $\{z_i := (\chi_i, k_i)\}_{i \in H}$ as the inputs of the honest clients in the outer MPC protocol to the external challenger.

During the watchlist correlations phase $\mathcal{B}$ intercepts the queries $\{K_{i,j}\}_{i \in M, j \in H}$ sent by corrupt parties and sets $C := \{x_{i,j}\}_{i \in M, j \in H}$. $\mathcal{B}$ instructs the external challenger to corrupt the set of clients given by M and the set of servers given by C.

The challenger provides $\{\phi_1^{i \to j}\}_{i \in H, j \in C}$. $\mathcal{B}$ 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 $|C'| \leq \lambda n^2$, then $\mathcal{B}$ instructs the external challenger to corrupt the servers indexed by C' and obtains $\{\phi_1^{i \to h}\}_{i \in H, h \in C'}$. For every $h \in [m] \backslash \{C \cup C'\}$, $\mathcal{B}$ sends $\{\phi_1^{i \to h}\}_{i \in M}$ as the first round messages generated by corrupted client to the honest server indexed by h. $\mathcal{B}$ obtains $\{\phi_2^h\}_{h \in [m] \backslash \{C \cup C'\}}$. $\mathcal{B}$ uses this to generate the final round message of the protocol as in Hyb_3.

On receiving the final round message from $\mathcal{A}$, $\mathcal{B}$ constructs the set C'' as in Hyb_3 and instructs the challenger to adaptively corrupt the servers in C'' if $|C''| \leq \lambda n^2$. It obtains $\{\phi^{i \to h}\}_{i \in H, h \in C''}$. $\mathcal{B}$ uses this to adaptively corrupt the honest parties in the inner protocol and obtain the secret key for those executions in $C' \cup C''$. Thus, $\mathcal{B}$ computes $\{\phi_2^h\}_{h \in [m]}$ as in Hyb_3. It runs out_Φ on the second round messages and computes $(y', \sigma'_1, \ldots, \sigma'_n)$. It then performs the MAC checks as in the description of the protocol. $\mathcal{B}$ runs D on the view of the adversary and the outputs of the honest parties and outputs whatever D outputs.

Since $|C| < 2\lambda n^2$ and $|C'| \leq \lambda n^2$, $|C''| \leq \lambda n^2$, the size of $|C \cup C' \cup C''| < 4\lambda n^2 < (m-1)/3$. 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. □

8.4 Instantiating the Inner Protocol

In this subsection, we give different instantiations of the inner protocol.

Using 2-sided Yao's protocol in the Two-party Case. In the case where there are only two parties, we observe that the two-sided Yao's protocol (where P_1 acts as the receiver in one side and P_2 acts as the receiver in the other side) in the OT correlations model satisfies all the required properties of the inner protocol. The first round message is equivocable since the receiver's message in an OT execution in the presence of OT correlations is equivocable.

Using [GIS18]. We can instantiate the inner protocol using the protocol given in [GIS18] in the OT-correlations model (where the correlations needed for this protocol can be generated using the watchlist functionality). We note that this protocol readily satisfies equivocality as a result of Property 7.4 and the observation that the receiver message of the information-theoretic OT protocol based on random OT correlations is equivocable. For the case where there are constant

- **Input:** P_1's input is $x_1, z_1 \in \mathbb{F}$ and P_2's input is $x_2, z_2 \in \mathbb{F}$.
- **OLE correlations:** Sample random a_1, a_2, b_1, b_2 from $\mathbb{F}$ such that $a_1 a_2 = b_1 + b_2$. P_1 gets (a_1, b_1) and P_2 gets (a_2, b_2).
- **Round-1:** P_1 sends $m_{1,1} = x_1 + a_1$ and P_2 sends $m_{2,1} = x_2 + a_2$.
- **Round-2:** P_1 sends $m_{1,2} = x_1(m_{2,1}) - z_1 + b_1$ and P_2 sends $m_{2,2} = x_2(m_{1,1}) - z_2 + b_2$.
- **Output Computation:** A party outputs $m_{1,1}m_{2,1} - m_{1,2} - m_{2,2}$.

Figure 13: Protocol for computing Degree-2 Polynomials taken verbatim from [LLW20].

number of parties, we can instead use standard λ-out-of-m OT correlations between each pair of parties rather than the complex watchlist correlations. For this, we can rely on the combinatorial result (see for instance, an argument given in Claim C.10) that if we choose random subsets $K_1, K_2, \ldots, K_n$ (for constant n) of $[m]$ where each set is of size λ, then with overwhelming probability, the size of their intersection is $O(\lambda)$.

Using [LLW20]. We now show that the protocol given in [LLW20] satisfies all the properties required by the inner protocol. We start with description of a protocol for computing degree-2 polynomials over a finite field $\mathbb{F}$ in Figure 13 from their work.

We now argue that the protocol described in Figure 13 satisfies Definition 8.1. Let us assume without loss of generality that P_1 is corrupted. In the correlations generation phase. the simulator samples random a_1, b_1 and provides it to the adversary. In the first round, the simulator samples a random $m_{2,1}$ and sends on behalf of the honest P_2. At the end of the first round, the adversary provides an input x_1, z_1 and the correlations (a'_1, b'_1) and the simulator checks if the first round message, $m_{1,1}$ is consistent with x_1 and the correlation given in the generation phase. We show that when adversary gives a correlation $(a'_1, b'_1) \neq (a_1, b_1)$ provided in the generation phase, then the probability that the honest party accepts this correlation is at most $1/|\mathbb{F}|$. This is because if the honest party accepts, then $(a'_1 - a_1)a_2 = (b'_1 - b_1)$ and this holds with probability $1/|\mathbb{F}|$ over the randomness of a_2. If the adversary's first round message is consistent, then the simulator obtains the output of the function and samples random $m_{2,2}$ such that $m_{1,2} + m_{2,2} - m_{2,1} \cdot m_{1,2} = x_1 x_2 + z_1 + z_2$ (where $m_{1,2}$ is honestly computed second round message of P_1). Otherwise, the simulator uses the input x_2, z_2 of P_2 and $a_2 = m_{2,1} - x_2$ and $b_2 = a_1 a_2 - b_1$ and computes $m_{2,2}$ honestly. It follows that except with probability $1/|\mathbb{F}|$ the view of the adversary when interacting with the simulator is identical to its view when interacting with the honest party.

In [LLW20], the first step is to construct an effective degree-2 MPRE [ABT18] in the presence of OLE correlations for arithmetic NC^1 circuits. To reduce the degree from 3 to 2, the paper used OLE correlations that are shared between two parties. Again, via an argument mentioned above, the probability that the adversary corrupting one of these two parties submits an OLE correlation that is different from the correlation given by the generator but forces an honest party to accept is at most $1/|\mathbb{F}|$. Then, the protocol described in Figure 13 is used to compute all the degree-2 terms in the MPRE. Thus, via a simple union bound, the view of the adversary when interacting with the honest parties is $\mathsf{poly}(|C|)/|\mathbb{F}|$-close to its view when interacting with the simulator.

8.5 Towards Improving the Concrete Efficiency

We now suggest methods to improve the concrete computational cost of the 2-party variant of the protocol. The concrete computational cost of this protocol is dominated by two parameters: (i) the concrete cost of the outer protocol, and (ii) the total amount of server computation across all instances of the inner protocol. We now elaborate on techniques to reduce both these costs.

If we settle for the weaker notion of *covert security* [AL07], which is good enough for many applications, then we need to only invoke the the inner protocol a constant number of times. Concretely, we can follow the approach of [LOP11] and have a watchlist that consists of only a single server, invoking the outer [IKP10] protocol with 7 servers. This gives a 2-sided NISC protocol where the inner protocol is invoked only 7 times and the resultant protocol has covert security with a constant probability of detecting cheaters.

A better approach, which does not require settling for a weaker notion of security, is to improve the amortized cost of the protocol from [IKP10]. The complexity of the original protocol scales cubically with the number of servers (a statistical security parameter in the 2-sided NISC application). This overhead comes from two sources: (1) using a *bivariate* polynomial for sharing each input (as in the BGW protocol [BGW88]), to support a reduction of a global consistency check to pairwise consistency checks; (2) using a multiparty conditional disclosure of secrets mechanism to enforce pairwise consistencies. Both kinds of overhead can be amortized over big computations. The first by encoding a quadratic number of inputs in a single bivariate polynomial, analogously to the standard share packing technique for the univariate case [FY92]. (This comes at the cost of slightly decreasing the corruption threshold.) The second overhead can be amortized by using a standard hybrid encryption technique. We leave a full exploration of this approach to future work, but given the success of similar ideas in leading to practical protocols [GMO16, AHIV17, CDG$^+$17, KKW18, HIMV19], we expect 2-sided NISC to follow a similar path.

9 Acknowledgments

We thank Saikrishna Badrinarayanan for useful discussions about this work.

Y. Ishai was supported by ERC Project NTSC (742754), NSF-BSF grant 2015782, BSF grant 2018393, and ISF grant 2774/20. D. Khurana was supported in part from a DARPA SIEVE award. A. Sahai was supported in part from a DARPA SIEVE award, NTT Research, NSF Frontier Award 1413955, BSF grant2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through Award HR00112020024 (for D. Khurana and A. Sahai). Work done in part when A. Srinivasan was at UC Berkeley and supported in part by AFOSR Award FA9550-19-1-0200, AFOSR YIP Award, NSF CNS Award 1936826, DARPA/ARL SAFEWARE Award W911NF15C0210, a Hellman Award and research grants by the Sloan Foundation, Okawa Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley). The views expressed are those of the authors and do not reflect the official policy or position of the funding agencies.

References

[AAG$^+$16] Divesh Aggarwal, Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Optimal computational split-state non-malleable codes. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 393–417, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[ABG$^+$20] Benny Applebaum, Zvika Brakerski, Sanjam Garg, Yuval Ishai, and Akshayaram Srinivasan. Separating two-round secure computation from oblivious transfer. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 71:1–71:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[ABT18] Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Perfect secure computation in two rounds. In *TCC 2018, Part I*, LNCS, pages 152–174. Springer, Heidelberg, Germany, March 2018.

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

[ADKO15] Divesh Aggarwal, Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Leakage-resilient non-malleable codes. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 398–426, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[ADN$^+$19] Divesh Aggarwal, Ivan Damgård, Jesper Buus Nielsen, Maciej Obremski, Erick Purwanto, João Ribeiro, and Mark Simkin. Stronger leakage-resilient and non-malleable secret sharing schemes for general access structures. LNCS, pages 510–539, Santa Barbara, CA, USA, 2019. Springer, Heidelberg, Germany.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 17*, pages 2087–2104, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.

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

[AJL$^+$12a] 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.

[AJL$^+$12b] 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, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 483–501. Springer, 2012.

[AL07] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In Salil P. Vadhan, editor, *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, volume 4392 of *Lecture Notes in Computer Science*, pages 137–156. Springer, 2007.

[AMPR14] Arash Afshar, Payman Mohassel, Benny Pinkas, and Ben Riva. Non-interactive secure computation based on cut-and-choose. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 387–404, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

[BCG$^+$19] 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.

[BD18] Zvika Brakerski and Nico Döttling. Two-message statistically sender-private OT from LWE. In *TCC 2018, Part II*, LNCS, pages 370–390. Springer, Heidelberg, Germany, March 2018.

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 479–488. ACM, 1996.

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

[BGJ$^+$18] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Amit Sahai. Promise zero knowledge and its applications to round optimal MPC. LNCS, pages 459–487, Santa Barbara, CA, USA, 2018. Springer, Heidelberg, Germany.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10. ACM, 1988.

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

[BL18] Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. LNCS, pages 500–532. Springer, Heidelberg, Germany, 2018.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *STOC*, pages 503–513, 1990.

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

[CDG$^+$17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In *Proceedings of the 2017 acm sigsac conference on computer and communications security*, pages 1825–1842, 2017.

[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.

[CGL16] Eshan Chattopadhyay, Vipul Goyal, and Xin Li. Non-malleable extractors and codes, with their many tampered extensions. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 285–298, Cambridge, MA, USA, June 18–21, 2016. ACM Press.

[COSV16] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Concurrent non-malleable commitments (and more) in 3 rounds. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 270–299, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[COSV17] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Four-round concurrent non-malleable commitments from one-way functions. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 127–157, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[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.

[DPW18] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. *J. ACM*, 65(4):20:1–20:32, 2018.

[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, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part I*, volume 11891 of *Lecture Notes in Computer Science*, pages 111–130. Springer, 2019.

[FY92] Matthew K. Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In S. Rao Kosaraju, Mike Fellows, Avi Wigderson, and John A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 699–710. ACM, 1992.

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press.

[GIS18] Sanjam Garg, Yuval Ishai, and Akshayaram Srinivasan. Two-round MPC: information-theoretic and black-box. In Amos Beimel and Stefan Dziembowski, editors, *Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part I*, volume 11239 of *Lecture Notes in Computer Science*, pages 123–151. Springer, 2018.

[GJK15] Vipul Goyal, Aayush Jain, and Dakshita Khurana. Witness signatures and non-malleable multi-prover zero-knowledge proofs. *IACR Cryptology ePrint Archive*, 2015:1095, 2015.

[GK96a] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *J. Cryptology*, 9(3):167–190, 1996.

[GK96b] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, 1996.

[GKP$^+$18] Vipul Goyal, Ashutosh Kumar, Sunoo Park, Silas Richelson, and Akshayaram Srinivasan. Non-malleable commitments from non-malleable extractors. Manuscript, accessed via personal communication, 2018.

[GLOV12] Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *53rd FOCS*, pages 51–60, New Brunswick, NJ, USA, October 20–23, 2012. IEEE Computer Society Press.

[GMO16] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. Zkboo: Faster zero-knowledge for boolean circuits. In *25th {usenix} security symposium ({usenix} security 16)*, pages 1069–1083, 2016.

[GMPP16] Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. In *EUROCRYPT*, pages 448–476, 2016.

[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.

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

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

[GPR16] Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. In *STOC*, pages 1128–1141, 2016.

[GRRV14] Vipul Goyal, Silas Richelson, Alon Rosen, and Margarita Vald. An algebraic approach to non-malleability. In *55th FOCS*, pages 41–50, Philadelphia, PA, USA, October 18–21, 2014. IEEE Computer Society Press.

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

[GSZ20] Vipul Goyal, Akshayaram Srinivasan, and Chenzhi Zhu. Multi-source non-malleable extractors and applications. Cryptology ePrint Archive, Report 2020/157, 2020. https://eprint.iacr.org/2020/157.

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

[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.

[HIMV19] Carmit Hazay, Yuval Ishai, Antonio Marcedone, and Muthuramakrishnan Venkitasubramaniam. Leviosa: Lightweight secure arithmetic computation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 327–344. ACM, 2019.

[HIV17] Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Actively secure garbled circuits with constant communication overhead in the plain model. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 3–39, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany.

[HK12] Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *Journal of Cryptology*, 25(1):158–193, January 2012.

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

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

[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.

[IKP$^+$16] Yuval Ishai, Eyal Kushilevitz, Manoj Prabhakaran, Amit Sahai, and Ching-Hua Yu. Secure protocol transformations. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 430–458. Springer, 2016.

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

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

[Kal05] Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 78–95, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany.

[Khu17] Dakshita Khurana. Round optimal concurrent non-malleability from polynomial hardness. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 139–171, Baltimore, MD, USA, November 12–15, 2017. Springer, Heidelberg, Germany.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31, Chicago, IL, USA, May 2–4, 1988. ACM Press.

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 525–537. ACM, 2018.

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

[KS17] Dakshita Khurana and Amit Sahai. How to achieve non-malleability in one or two rounds. In *58th FOCS*, pages 564–575. IEEE Computer Society Press, 2017.

[LLW20] Huijia Lin, Tianren Liu, and Hoeteck Wee. Information-theoretic 2-round MPC without round collapsing: Adaptive security, and more. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part II*, volume 12551 of *Lecture Notes in Computer Science*, pages 502–531. Springer, 2020.

[LOP11] Yehuda Lindell, Eli Oxman, and Benny Pinkas. The IPS compiler: Optimizations, variants and concrete efficiency. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 259–276, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany.

[MR17] Payman Mohassel and Mike Rosulek. Non-interactive secure 2PC in the offline/online and batch settings. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 425–455, Paris, France, May 8–12, 2017. Springer, Heidelberg, Germany.

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

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

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

[PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *FOCS*, pages 366–375, 2002.

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

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

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press.

A Secure Multiparty Computation

Here, we provide a formal definition of secure multiparty computation. Parts of this section have been taken verbatim from [Gol04].

A multi-party protocol is cast by specifying a random process that maps pairs of inputs to pairs of outputs (one for each party). We refer to such a process as a functionality. The security of a protocol is defined with respect to a functionality f. In particular, let n denote the number of parties. A non-reactive n-party functionality f is a (possibly randomized) mapping of n inputs to n outputs. A multiparty protocol with security parameter λ for computing a non-reactive functionality f is a protocol running in time $\mathsf{poly}(\lambda)$ and satisfying the following correctness requirement: if parties $P_1, \ldots, P_n$ with inputs $(x_1, \ldots, x_n)$ respectively, all run an honest execution of the protocol, then the joint distribution of the outputs $y_1, \ldots, y_n$ of the parties is statistically close to $f(x_1, \ldots, x_n)$. A reactive functionality f is a sequence of non-reactive functionalities $f = (f_1, \ldots, f_\ell)$ computed in a stateful fashion in a series of phases. Let x_i^j denote the input of P_i in phase j, and let s^j denote the state of the computation after phase j. Computation of f proceeds by setting s^0 equal to the empty string and then computing $(y_1^j, \ldots, y_n^j, s^j) \leftarrow f_j(s^{j-1}, x_1^j, \ldots, x_n^j)$ for $j \in [\ell]$, where y_i^j denotes the output of P_i at the end of phase j. A multi-party protocol computing f also runs in ℓ phases, at the beginning of which each party holds an input and at the end of which each party obtains an output. (Note that parties may wait to decide on their phase-j input until the beginning of that phase.) Parties maintain state throughout the entire execution. The correctness requirement is that, in an honest execution of the protocol, the joint distribution of all the outputs $\{y_1^j, \ldots, y_n^j\}_{j=1}^{\ell}$ of all the phases is statistically close to the joint distribution of all the outputs of all the phases in a computation of f on the same inputs used by the parties.

A.1 Defining Security.

We assume that readers are familiar with standard simulation-based definitions of secure multiparty computation in the standalone setting. We provide a self-contained definition for completeness and refer to [Gol04] for a more complete description. The security of a protocol (with respect to a functionality f) is defined by comparing the real-world execution of the protocol with an ideal-world evaluation of f by a trusted party. More concretely, it is required that for every adversary $\mathcal{A}$, which attacks the real execution of the protocol, there exist an adversary Sim, also referred to as a simulator, which can *achieve the same effect* in the ideal-world. Let's denote $\mathbf{x} = (x_1, \ldots, x_n)$.

The real execution In the real execution of the n-party protocol π for computing f is executed in the presence of an adversary $\mathcal{A}$. The honest parties follow the instructions of π. The adversary $\mathcal{A}$ takes as input the security parameter k, the set $I \subset [n]$ of corrupted parties, the inputs of the corrupted parties, and an auxiliary input z. $\mathcal{A}$ sends all messages in place of corrupted parties and may follow an arbitrary polynomial-time strategy.

The interaction of $\mathcal{A}$ with a protocol π defines a random variable $\mathsf{REAL}_{\pi,\mathcal{A}(z),I}(\lambda, \mathbf{x})$ whose value is determined by the coin tosses of the adversary and the honest players. This random variable contains the output of the adversary (which may be an arbitrary function of its view) as well as the outputs of the uncorrupted parties. We let $\mathsf{REAL}_{\pi,\mathcal{A}(z),I}$ denote the distribution ensemble $\{\mathsf{REAL}_{\Pi,\mathcal{A}(z),I}(\lambda, \mathbf{x})\}_{k\in\mathsf{N},\langle\mathbf{x},z\rangle\in\{0,1\}^*}$.

The ideal execution – security with abort. In this second variant of the ideal model, fairness and output delivery are no longer guaranteed. This is the standard relaxation used when a strict majority of honest parties is not assumed. In this case, an ideal execution for a function f proceeds as follows:

- **Send inputs to the trusted party:** As before, the parties send their inputs to the trusted party, and we let x'_i denote the value sent by P_i. Once again, for a semi-honest adversary we require $x'_i = x_i$ for all $i \in I$.
- **Trusted party sends output to the adversary:** The trusted party computes $f(x'_1, \ldots, x'_n) = (y_1, \ldots, y_n)$ and sends $\{y_i\}_{i\in I}$ to the adversary.
- **Adversary instructs trusted party to abort or continue:** This is formalized by having the adversary send either a continue or abort message to the trusted party. (A semi-honest adversary never aborts.) In the latter case, the trusted party sends to each uncorrupted party P_i its output value y_i. In the former case, the trusted party sends the special symbol $\perp$ to each uncorrupted party.
- **Outputs:** Sim outputs an arbitrary function of its view, and the honest parties output the values obtained from the trusted party.

The interaction of Sim with the trusted party defines a random variable $\mathsf{IDEAL}_{f,\mathcal{A}(z)}(\lambda, \mathbf{x})$ as above,and we let $\{\mathsf{IDEAL}_{f,\mathcal{A}(z),I}(\lambda, \mathbf{x})\}_{k\in\mathsf{N},\langle\mathbf{x},z\rangle\in\{0,1\}^*}$. Having defined the real and the ideal worlds, we now proceed to define our notion of security.

Definition A.1. *Let k be the security parameter. Let f be an n-party randomized functionality, and Π be an n-party protocol for $n \in \mathsf{N}$. We say that Π t-securely computes f in the presence of malicious (resp., semi-honest) adversaries if for every PPT adversary (resp., semi-honest adversary) $\mathcal{A}$ there exists a PPT adversary (resp., semi-honest adversary)* Sim *such that for any $I \subset [n]$ with $|I| \leq t$ the following quantity is negligible:*

$$|Pr[\mathsf{REAL}_{\Pi,\mathcal{A}(z),I}(\lambda, \mathbf{x}) = 1] - Pr[\mathsf{IDEAL}_{f,\mathcal{A}(z),I}(\lambda, \mathbf{x}) = 1]|$$

where $\mathbf{x} = \{x_i\}_{i\in[n]} \in \{0,1\}^*$ and $z \in \{0,1\}^*$.

Remark A.2 (Security with Selective Abort)**.** *We can consider a slightly weaker definition of security where the ideal world adversary can instruct the trusted party to send aborts to a subset of the uncorrupted parties. For the rest of the uncorrupted parties, it instructs the trusted functionality to deliver their output. This weakened definition is called as security with selective abort.*

Remark A.3 (Privacy with Knowledge of Outputs)**.** *Ishai et al. [IKP10] considered a further weakening of the security definition where the trusted party first delivers the output to the ideal world adversary which then provides an output to be delivered to all the honest parties. They called this security notion as privacy with knowledge of outputs and showed a transformation from this notion to security with selective abort using unconditional MACs.*

A.2 Security Against Semi-Malicious Adversaries

We take this definition almost verbatim from [AJL$^+$12b]. We consider a notion of a semi-malicious adversary that is stronger than the standard notion of semi-honest adversary and formalize security against semi-malicious adversaries. A semi-malicious adversary is modeled as an interactive Turing machine (ITM) which, in addition to the standard tapes, has a special witness tape. In each round of the protocol, whenever the adversary produces a new protocol message msg on behalf of some party P_k, it must also write to its special witness tape some pair (x, r) of input x and randomness r that explains its behavior. More specifically, all of the protocol messages sent by the adversary on behalf of P_k up to that point, including the new message m, must exactly match the honest protocol specification for P_k when executed with input x and randomness r. Note that the witnesses given in different rounds need not be consistent. Also, we assume that the attacker is rushing and hence may choose the message m and the witness (x, r) in each round adaptively, after seeing the protocol messages of the honest parties in that round (and all prior rounds). Lastly, the adversary may also choose to abort the execution on behalf of P_k in any step of the interaction.

Definition A.4. *We say that a protocol* Π *securely realizes* f *for semi-malicious adversaries if it satisfies Definition A.1 when we only quantify over all semi-malicious adversaries* A*.*

B Special Oblivious Transfer Protocol

In this section, we give a construction of the special two-round oblivious transfer protocol from Section 7.2. This construction makes black-box use of a two-round oblivious transfer $(\mathsf{OT}'_1, \mathsf{OT}'_2, \mathsf{OT}'_3)$ with security against semi-malicious adversaries. We give the construction below.

- $\mathsf{OT}_1(1^\lambda, b)$: Choose random strings $r_0, r_1 \leftarrow \{0,1\}^*$ and compute $\mathsf{otm}'_{b,1} \leftarrow \mathsf{OT}'_1(1^\lambda, 0; r_0)$ and $\mathsf{otm}'_{1-b,1} \leftarrow \mathsf{OT}'_1(1^\lambda, 1; r_1)$. Set $\mathsf{otm}_1 := (\mathsf{otm}'_{0,1}, \mathsf{otm}'_{1,1})$ and $\omega := r_b$.
- $\mathsf{OT}_2(\mathsf{otm}_1, m_0, m_1)$: For each $b \in \{0,1\}$, generate $\mathsf{otm}'_{b,2} \leftarrow \mathsf{OT}'_2(\mathsf{otm}'_{b,1}, (m_b, \perp))$. Set $\mathsf{otm}_2 := (\mathsf{otm}'_{0,2}, \mathsf{otm}'_{1,2})$.
- $\mathsf{OT}_3(\mathsf{otm}_2, (b, \omega))$: Run $\mathsf{OT}'_3(\mathsf{otm}'_{b,2}, (0, \omega))$ to recover m_b.

Correctness follows directly from the correctness of the protocol $(\mathsf{OT}'_1, \mathsf{OT}'_2, \mathsf{OT}'_3)$. We now show the security properties:

- **Equivocal Receiver Security.** The equivocal simulator $\mathsf{Sim}^{eq}_{\mathsf{OT}}$ generates otm'_0 and otm'_1 as $\mathsf{OT}'_1(1^\lambda, 0)$. The indistinguishability follows directly from the receiver security of OT' which guarantees that $\mathsf{OT}'_1(1^\lambda, 0) \approx_c \mathsf{OT}'_1(1^\lambda, 1)$.

- **Security in the No Corruption Setting.** We consider a couple of intermediate hybrids.
  - Hyb_0 : Distribution of otm_1 and otm_2 when the inputs are b and m_0, m_1 respectively.
  - Hyb_1 : In this hybrid, we change $\mathsf{otm}'_{b,1}$ in otm_2 to $\mathsf{OT}'_1(1^\lambda, 1)$. This hybrid is computationally indistinguishable to the previous hybrid from the receiver security of OT'.
  - Hyb_2 : In this hybrid, we change the inputs used in generating both $\mathsf{otm}'_{0,2}$ and $\mathsf{otm}'_{1,2}$ to $(\perp, \perp)$. This hybrid is computationally close to the previous one from the sender security of OT' against semi-malicious adversaries.

  Via an identical argument, we can show that Hyb_2 is computationally indistinguishable to a distribution of otm_1 and otm_2 when the inputs are b' and m'_0, m'_1 respectively.

- **Sender Security.** This follows directly from the semi-malicious security property of OT'.

C Rewinding Secure Two-party Computation

In this section, we will describe a 4-round, black-box two-party computation protocol for NC^1 functions that satisfies 1-rewind sender security (see Definition 4.3). This is used as a building block in constructing the watchlist protocol in Section 5.

Theorem C.1. *Assume black-box access to a public key encryption with pseudorandom public keys. There exists a four-round protocol Π that is 1-rewinding sender secure with delayed function selection for NC^1 circuits (see Definition 4.3).*

C.1 Construction

In this subsection, we describe our construction of a 1-rewinding sender secure computation protocol with delayed function selection.

Building Blocks. We use the following building blocks in our construction.

- A two-round information-theoretic 2 client, m server MPC protocol for computing NC^1 circuits and satisfying security with selective abort against a malicious, adaptive adversary corrupting 1 client and at most $t = (m-1)/3$ servers from [IKP10, Pas12]. Here, we fix $m = 8\lambda$. We need the protocol to satisfy the following two properties which can be added to the protocol from [IKP10, Pas12]:
  - We require the first round message of the protocol to be independent of the function description and only depends on its size. We note that this property can be generically added to any two-round client server MPC protocol. Specifically, assume the existence of a virtual client who holds the description of the function, uses some default randomness to generate the first round message using the description of the function as input. All the other clients send their messages to the servers using their input for computing the universal circuit. Since, the virtual client uses some default randomness, the messages from this client can be emulated by the servers.
  - Given a first round message from the clients, the servers in the second round, are given descriptions of two functions f_0, f_1 and should be able to generate the second round message corresponding to these two functions on the same first round message from

the client. In other words, we require the protocol's first round message to be reusable once. We observe that this additional requirement can be added to the protocol of [IKP10, Pas12]. Specifically, the first round message from the client comprises of three parts, a secret sharing of its input, a random secret sharing of 0 and a multiparty conditional disclosure of secrets (MCDS) message. We note that the secret sharing of its input can be reused whereas the random secret sharing of 0 and the MCDS message cannot be reused. For this purpose, instead of generating a single MCDS message and a random secret sharing of 0, we generate $2 \times |f|$ of them in the first round, where $|f|$ denotes the size of the function description. In the second round, the servers use the description of the function and for each $i \in [|f|]$, they use MCDS message and the secret sharing of 0 corresponding to the bit f_i. Specifically, the server adds all the chosen secret sharings of 0 to obtain a single share and it adds all the chosen MCDS secrets to obtain a single secret. It then generates the second round MCDS message for each of the chosen indices. This ensures that if $f_0 \neq f_1$, there exists at least one index $i \in [|f|]$ such that the MCDS message and the random secret sharing of 0 is only used once in the two server executions.

- A 4-round, 1-out-of-2 parallel-composable[14] OT protocol $\mathsf{OT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3, \mathsf{OT}_4, \mathsf{out}_{\mathsf{OT}})$ that satisfies the receiver security as described in Definition 4.3 and a slightly modified 1-rewinding sender security. The first difference is that there is no delayed function selection as the function to be computed is that of an oblivious transfer (i.e., $f_0 = f_1 = \mathsf{OT}$). The second difference is that in Expt_2 given in Figure 4, $\mathsf{Sim}^1_{\mathsf{OT},\mathcal{S}}$ outputs the receiver's choice bits along with the first round OT messages and sk_{OT}. We will call this protocol as a 1-out-of-2 parallel-composable OT with 1-rewinding sender security. In Appendix D, we give a construction of this primitive that makes black-box use of any public key encryption with pseudorandom public keys.
- A pseudorandom function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^*$ and a symmetric key encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.
- A 1-rewinding secure extractable commitment scheme $(\mathsf{ECom}_1, \mathsf{ECom}_2, \mathsf{ECom}_3)$ from Section 3.3.
- A garbling scheme $(\mathsf{Garble}, \mathsf{Eval})$.

We show the following Theorem which implies Theorem C.1 as a corollary of Theorem D.1.

Theorem C.2. *Assume black-box access to a 4-round, 1-out-of-2 parallel composable OT protocol with 1-rewinding sender security. Then there exists a four-round protocol* Π *that is 1-rewinding sender secure with delayed function selection for* NC^1 *circuits (see Definition 4.3).*

Description of the Protocol. We give the formal description of the protocol below.

- **Round-1:** In the first round, $\mathcal{R}$ does the following:
    1. It computes $(x_1, \ldots, x_m) \leftarrow \Phi_1(x_{\mathcal{R}})$. For each $i \in [m]$, we assume that $x_i \in \{0,1\}^\ell$ and let $x_{i,j}$ denote the j-th bit of x_i for each $j \in [\ell]$.

[14]A parallel-composable OT is one that remains secure when the sender and receiver invoke (unbounded) polynomially many executions in parallel. Unlike the watchlist protocol, in each of these executions the parties which play the role of the sender and the receiver are the same.

2. For each $i \in [m]$ and $j \in [\ell]$, it chooses λ bits $x_{i,j,1}, \ldots, x_{i,j,\lambda}$ such that $\oplus_{k\in[\lambda]} x_{i,j,k} = x_{i,j}$.
3. For each $i \in [m]$, $j \in [\ell]$ and $k \in [\lambda]$, it computes $\mathsf{ots}_1^{i,j,k} := \mathsf{OT}_1(1^\lambda, x_{i,j,k})$.
4. It chooses a random subset $K \subset [m]$ of size λ and let $(K_1, \ldots, K_m) \in \{0,1\}^m$ denote its characteristic vector.
5. It computes $\mathsf{ots}_1^i := \mathsf{OT}_1(1^\lambda, K_i)$ for each $i \in [m]$.
6. It sends $\{\mathsf{ots}_1^i\}_{i\in[m]}, \{\mathsf{ots}_1^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to the sender.

- **Round-2:** The sender $\mathcal{S}$ does the following.
  1. It computes $(y_1, \ldots, y_m) \leftarrow \Phi_1(x_{\mathcal{S}})$.
  2. For each $i \in [m]$, $j \in [\ell]$ and $k \in [\lambda]$,
     (a) It samples $sk_0^{i,j,k}, sk_1^{i,j,k}$ from $\mathsf{KeyGen}(1^\lambda)$.
     (b) It computes $\mathsf{otm}_2^{i,j,k} \leftarrow \mathsf{OT}_2(\mathsf{otm}_1^{i,j,k}, sk_0^{i,j,k}, sk_1^{i,j,k})$.
  3. It chooses random strings $s_1, \ldots, s_m$ uniformly from $\{0,1\}^*$.
  4. For each $i \in [m]$,
     (a) It samples sk_0^i, sk_1^i from $\mathsf{KeyGen}(1^\lambda)$.
     (b) It computes $\mathsf{otm}_2^i \leftarrow \mathsf{OT}_2(\mathsf{otm}_1^i, \mathsf{sk}_0^i, \mathsf{sk}_1^i)$.
     (c) It chooses a random PRF key $\mathsf{key}_i \leftarrow \{0,1\}^*$.
     (d) It computes $\mathsf{Com}_1^i := \mathsf{ECom}_1(1^\lambda, (y_i, \mathsf{key}_i, \{sk_0^{i,j,k}, sk_1^{i,j,k}\}_{j\in[\ell],k\in[\lambda]}); s_i)$.
  5. It sends $\{\mathsf{Com}_1^i, \mathsf{ots}_2^i\}_{i\in[m]}, \{\mathsf{ots}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to $\mathcal{R}$.

- **Round-3**: In the third round, $\mathcal{R}$ does the following:
  1. For each $i \in [m]$, $j \in [\ell]$ and $k \in [\lambda]$, it computes $\mathsf{otm}_3^{i,j,k} \leftarrow \mathsf{OT}_3(\mathsf{otm}_2^{i,j,k}, x_{i,j,k})$.
  2. For each $i \in [m]$,
     (a) It computes $\mathsf{otm}_3^i \leftarrow \mathsf{OT}_3(\mathsf{otm}_2^i, K_i)$.
     (b) It samples Com_2^i using $\mathsf{ECom}_2(\mathsf{Com}_1^i)$.
  3. It sends $\{\mathsf{Com}_2^i, \mathsf{ots}_3^i\}_{i\in[m]}, \{\mathsf{ots}_3^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to $\mathcal{S}$.

- **Round-4:** In the final round, $\mathcal{S}$ does the following:
  1. For each $i \in [m]$,
     (a) It computes $\mathsf{otm}_4^i \leftarrow \mathsf{OT}_4(\mathsf{otm}_1^i, \mathsf{otm}_3^i, (sk_0^i, sk_1^i))$.
     (b) It computes $\widetilde{C}, \{\mathsf{lab}_b^i\}_{i\in[m],b\in\{0,1\}} \leftarrow \mathsf{Garble}(1^\lambda, C[s_1, \ldots, s_m])$ where C checks if the input $(K_1, \ldots, K_m)$ has hamming weight exactly k and if it is the case, it outputs $\{s_i\}_{i:K_i=1}$.
     (c) For each $b \in \{0,1\}$, it computes $ct_b^i := \mathsf{Enc}(sk_b^i, \mathsf{lab}_b^i)$.
     (d) It computes $\mathsf{Com}_3^i := \mathsf{ECom}_3(\mathsf{Com}_2^i, (y_i, \mathsf{key}_i, \{sk_0^{i,j,k}, sk_1^{i,j,k}\}_{j\in[\ell],k\in[\lambda]}); s_i)$.
     (e) Let $\Phi_{2,i}$ be the function that takes $\{y_{i,j}\}_{j\in[\ell]}$ and $\{x_{i,j,k}\}_{j\in[\ell],k\in[\lambda]}$ as inputs and first reconstructs $\{x_{i,j}\}_{j\in[\ell]}$ and then applies the function computed by the i-th server in the outer protocol Φ on $f, \{x_{i,j}, y_{i,j}\}_{j\in[\ell]}$.
     (f) It computes $\widetilde{\Phi}_{2,i}, \{\overline{\mathsf{lab}}_0^{i,j}, \overline{\mathsf{lab}}_1^{i,j}\}_{j\in[\ell]}, \{\mathsf{lab}_0^{i,j,k}, \mathsf{lab}_1^{i,j,k}\}_{j\in[\ell],k\in[\lambda]} \leftarrow \mathsf{Garble}(1^\lambda, \Phi_{2,i}; r_i)$ where $r_i := \mathsf{PRF}_{\mathsf{key}_i}(f)$. Here, $\{\overline{\mathsf{lab}}_0^{i,j}, \overline{\mathsf{lab}}_1^{i,j}\}_{j\in[\ell]}$ are the input labels for the sender and $\{\mathsf{lab}_0^{i,j,k}, \mathsf{lab}_1^{i,j,k}\}_{j\in[\ell],k\in[\lambda]}$ are the input labels for the receiver.

2. For each $i \in [m]$, $j \in [\ell]$ and $k \in [\lambda]$,
    (a) It computes $\mathsf{otm}_4^{i,j,k} \leftarrow \mathsf{OT}_4(\mathsf{otm}_1^{i,j,k}, \mathsf{otm}_3^{i,j,k}, (sk_0^{i,j,k}, sk_1^{i,j,k}))$.
    (b) It samples two PRF keys $\mathsf{key}_0^{i,j,k}, \mathsf{key}_1^{i,j,k}$.
    (c) It computes $ct_b^{i,j,k} := \mathsf{Enc}(sk_b^{i,j,k}, \mathsf{lab}_b^{i,j,k}; \mathsf{PRF}_{\mathsf{key}_b^{i,j,k}}(f))$ for each $b \in \{0,1\}$.
3. It sends $\widetilde{C}, \{ct_0^i, ct_1^i, \mathsf{Com}_3^i, \mathsf{ots}_4^i, \widetilde{\Phi}_{2,i}\}_{i\in[m]}$ and $\{\mathsf{otm}_4^{i,j,k}, \overline{\mathsf{lab}}_{y_{i,j}}^{i,j}, ct_0^{i,j,k}, ct_1^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to $\mathcal{R}$.

- **Output:** To compute the output, $\mathcal{R}$ does the following:
    1. It recovers $sk_{K_i}^i$ for each $i \in [m]$ from $\mathsf{otm}_2^i, \mathsf{otm}_4^i$ using the input K_i and the random tape used in generating $\mathsf{otm}_1^i, \mathsf{otm}_3^i$. It then computes $\mathsf{lab}_{K_i}^i := \mathsf{Dec}(sk_{K_i}^i, ct_{K_i}^i)$.
    2. It computes $\{s_i\}_{i\in K} \leftarrow \mathsf{Eval}(\widetilde{C}, \{\mathsf{lab}_{K_i}^i\}_{i\in[m]})$.
    3. For each $i \in [m]$, $j \in [\ell]$ and $k \in [\lambda]$,
        (a) It computes $sk_{x_{i,j,k}}^{i,j,k}$ from $\mathsf{otm}_2^{i,j,k}$ and $\mathsf{otm}_4^{i,j,k}$ using the input $x_{i,j,k}$ and the random tape used in generating $\mathsf{otm}_1^{i,j,k}, \mathsf{otm}_3^{i,j,k}$.
        (b) It computes $\mathsf{lab}_{x_{i,j,k}}^{i,j,k} := \mathsf{Dec}(sk_{x_{i,j,k}}^{i,j,k}, ct_{x_{i,j,k}}^{i,j,k})$.
    4. For each $i \in K$,
        (a) It recovers $(y_i, \mathsf{key}_i, \{sk_0'^{i,j,k}, sk_1'^{i,j,k}\}_{j\in[\ell]})$ from $(\mathsf{Com}_1^i, \mathsf{Com}_3^i)$ and randomness s_i.
        (b) It computes $r_i \leftarrow \mathsf{PRF}_{\mathsf{key}_i}(f)$.
        (c) It computes $\widetilde{\Phi}_{2,i}, \{\widetilde{\mathsf{lab}}_0^{i,j}, \widetilde{\mathsf{lab}}_1^{i,j}\}_{j\in[\ell]}, \{\mathsf{lab}_0'^{i,j,k}, \mathsf{lab}_1'^{i,j,k}\}_{j\in[\ell]} \leftarrow \mathsf{Garble}(1^\lambda, \Phi_{2,i}; r_i)$.
        (d) It checks if $\widetilde{\Phi}_{2,i}$ that is received in the fourth round is the same as the one computed above.
        (e) For each $j \in [\ell]$,
            i. It checks if $\overline{\mathsf{lab}}_{y_{i,j}}^{i,j} = \widetilde{\mathsf{lab}}_{y_{i,j}}^{i,j}$.
            ii. For each $k \in [\lambda]$, it checks if $sk_{x_{i,j,k}}^{i,j,k} = sk_{x_{i,j,k}}'^{i,j,k}$.
            iii. For each $b \in \{0,1\}$, it computes $\mathsf{lab}_{1-x_{i,j,k}}^{i,j,k} := \mathsf{Dec}(sk_{1-x_{i,j,k}}'^{i,j,k}, ct_{1-x_{i,j,k}}^{i,j,k})$.
            iv. For each $b \in \{0,1\}$ and $k \in [\lambda]$, it checks if $\mathsf{lab}_b^{i,j,k} = \mathsf{lab}_b'^{i,j,k}$.
    5. If any of the checks fail, then it aborts.
    6. For each $i \in [m]$, it computes $z_i \leftarrow \mathsf{Eval}(\widetilde{\Phi}_{2,i}, \{\overline{\mathsf{lab}}_{y_{i,j}}^{i,j}\}_{j\in[\ell]}, \{\mathsf{lab}_{x_{i,j,k}}^{i,j,k}\}_{j\in[\ell],k\in[\lambda]})$.
    7. It finally computes out by running out_Φ on $\{z_i\}_{i\in[m]}$, the input $x_{\mathcal{R}}$ and the random tape used in generating $(x_1, \ldots, x_m)$.

C.2 Proof of Security

In this subsection, we show that the above described protocol satisfies Definition 4.3. We start with the 1-rewinding sender security property.

C.2.1 1-Rewinding Sender Security

We begin with the descriptions of simulators $(\mathsf{Sim}_{\mathcal{S}}^1, \mathsf{Sim}_{\mathcal{S}}^2)$.

Description of $\mathsf{Sim}^1_{\mathcal{S}}$.

1. $\mathsf{Sim}^1_{\mathcal{S}}$ constructs an adversary $\mathcal{A}_1$ that interacts with $\mathcal{A}$ internally and receives the messages corresponding to the first three rounds of the OT protocol externally. Specifically, $\mathcal{A}_1$ does the following:

   - $\mathcal{A}_1$ initializes $\mathcal{A}$ and obtains the first round messages $\{\mathsf{otm}_1^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{\mathsf{otm}_1^i\}_{i\in[m]}$ from $\mathcal{A}$. It forwards these messages externally.
   - $\mathcal{A}_1$ receives $\{\mathsf{otm}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{\mathsf{otm}_2^i\}_{i\in[m]}$ externally.
   - For each $i \in [m]$, $\mathcal{A}_1$ generates $\mathsf{Com}_1^i \leftarrow \mathsf{ECom}_1(1^\lambda, \bot)$.
   - $\mathcal{A}_1$ now runs $\mathcal{A}$ on $\{\mathsf{otm}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{\mathsf{Com}_1^i, \mathsf{otm}_2^i\}_{i\in[m]}$.
   - $\mathcal{A}_1$ receives $\{f_b\}_{b\in\{0,1\}}, \{\mathsf{otm}_3^{i,j,k}[b]\}_{i\in[m],j\in[\ell],k\in[\lambda],b\in\{0,1\}}, \{\mathsf{Com}_2^i[b], \mathsf{otm}_3^i[b]\}_{i\in[m],b\in\{0,1\}}$ from $\mathcal{A}$ and forwards $\{\mathsf{otm}_3^{i,j,k}[b]\}_{i\in[m],j\in[\ell],k\in[\lambda],b\in\{0,1\}}, \{\mathsf{otm}_3^i[b]\}_{i\in[m],b\in\{0,1\}}$ externally.

2. $\mathsf{Sim}^1_{\mathcal{S}}$ now runs $\mathsf{Sim}^1_{\mathsf{OT},\mathcal{S}}$ on $\mathcal{A}_1$ and obtains $x_{\mathsf{OT}}, sk_{\mathsf{OT}}$ and $\{\mathsf{otm}_1^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{\mathsf{otm}_1^i\}_{i\in[m]}$.

3. If x_{OT} is the special symbol abort, then

   (a) $\mathsf{Sim}^1_{\mathcal{S}}$ sets $x_{\mathcal{R}}$ to be abort and sets sk to be the aborting transcript of the first three rounds that is obtained from the random tape of $\mathsf{Sim}^1_{\mathsf{OT},\mathcal{S}}$ and the random tape of $\mathcal{A}_1$.

4. If $x_{\mathsf{OT}} = (\{x_{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{K_i\}_{i\in[m]})$, then

   (a) $\mathsf{Sim}^1_{\mathcal{S}}$ runs independent executions of $\mathsf{Sim}^1_{\mathsf{OT},\mathcal{S}}$ on $\mathcal{A}_1$ until it obtains 12λ executions where x_{OT} is not equal to the special symbol abort. Let T be the total number of executions needed. $\mathsf{Sim}^1_{\mathcal{S}}$ sets $\widetilde{\epsilon} = 12\lambda/T$.

   (b) $\mathsf{Sim}^1_{\mathcal{S}}$ sets $sk = (sk_{\mathsf{OT}}, \widetilde{\epsilon}, x_{\mathsf{OT}})$ and $\pi_1 = \{\mathsf{otm}_1^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{\mathsf{otm}_1^i\}_{i\in[m]}$.

5. $\mathsf{Sim}^1_{\mathcal{S}}$ maintains an internal step counter, and if it runs for more than 2^λ time steps then it aborts and outputs a special symbol $\mathsf{timeout}$.

Description of $\mathsf{Sim}^2_{\mathcal{S}}$.

1. If x_{OT} corresponds to the special symbol abort, then $\mathsf{Sim}^2_{\mathcal{S}}$ outputs sk.

2. Else, it parses x_{OT} as $(\{x_{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{K_i\}_{i\in[m]})$.

3. Let K be the set of all $i \in [m]$ such $K_i = 1$. If $K > \lambda$, then we reset $K = \emptyset$.

4. $\mathsf{Sim}^2_{\mathcal{S}}$ now runs Sim_{Φ} by corrupting the client corresponding to the receiver and the set of servers indexed by K and obtains $\{y_i\}_{i\in K}$.

5. For each $i \in K$, $\mathsf{Sim}^2_{\mathcal{S}}$ chooses a PRF key $\mathsf{key}_i \leftarrow \{0,1\}^*$. Additionally, for each $i \in [m]$, it chooses sk_0^i, sk_1^i from $\mathsf{KeyGen}(1^\lambda)$ and for each $j \in [\ell]$ and $k \in [\lambda]$, it samples $sk_0^{i,j,k}, sk_1^{i,j,k}$ from $\mathsf{KeyGen}(1^\lambda)$.

6. $\mathsf{Sim}^2_{\mathcal{S}}$ now constructs an adversary $\mathcal{A}_2$ that interacts with $\mathcal{A}$ internally and obtains the messages of the last three rounds of the OT protocol externally. Specifically, $\mathcal{A}_2$ does the following:

- $\mathcal{A}_2$ receives $\{\mathsf{otm}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{\mathsf{otm}_2^i\}_{i\in[m]}$ externally.
- For each $i \in K$, $\mathcal{A}_2$:
  - Samples $s_i \leftarrow \{0,1\}^*$.
  - Computes $\mathsf{Com}_1^i := \mathsf{ECom}_1(1^\lambda, (y_i, \mathsf{key}_i, \{sk_0^{i,j,k}, sk_1^{i,j,k}\}_{j\in[\ell],k\in[\lambda]}); s_i)$.
- For each $i \notin K$, $\mathcal{A}_2$:
  - Computes $\mathsf{Com}_1^i \leftarrow \mathsf{ECom}_1(1^\lambda, \bot)$.
- $\mathcal{A}_2$ runs $\mathcal{A}$ on $\{\mathsf{otm}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{\mathsf{Com}_1^i, \mathsf{otm}_2^i\}_{i\in[m]}$.
- $\mathcal{A}_1$ receives $\{f_b\}_{b\in\{0,1\}}, \{\mathsf{otm}_3^{i,j,k}[b]\}_{i\in[m],j\in[\ell],k\in[\lambda],b\in\{0,1\}}, \{\mathsf{Com}_2^i[b], \mathsf{otm}_3^i[b]\}_{i\in[m],b\in\{0,1\}}$ from $\mathcal{A}$ and forwards $\{\mathsf{otm}_3^{i,j,k}[b]\}_{i\in[m],j\in[\ell],k\in[\lambda],b\in\{0,1\}}, \{\mathsf{otm}_3^i[b]\}_{i\in[m],b\in\{0,1\}}$ externally.

7. $\mathsf{Sim}_{\mathcal{S}}^2$ now runs $\mathsf{Sim}_{\mathsf{OT},\mathcal{S}}^2$ on $\mathcal{A}_2$ with the input $\{sk_{x_{i,j,k}}^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ and $\{sk_{K_i}^i\}_{i\in[m]}$ for $\lambda^2/\widetilde{\epsilon}$ times (using independent random tapes). If in each of these executions, $\mathsf{Sim}_{\mathsf{OT},\mathcal{S}}^2$ outputs the error symbol fail, then $\mathsf{Sim}_{\mathcal{S}}^1$ outputs the error symbol fail and aborts.
8. Otherwise, let $\{f_b\}_{b\in\{0,1\}}$, $\{\mathsf{otm}_3^{i,j,k}[b]\}_{i\in[m],j\in[\ell],k\in[\lambda],b\in\{0,1\}}, \{\mathsf{Com}_2^i[b], \mathsf{otm}_3^i[b]\}_{i\in[m],b\in\{0,1\}}$ be the third round message and let $\{\mathsf{otm}_4^{i,j,k}[b]\}_{i\in[m],j\in[\ell],k\in[\lambda],b\in\{0,1\}}, \{\mathsf{otm}_4^i[b]\}_{i\in[m],b\in\{0,1\}}$ be the final round message output by $\mathsf{Sim}_{\mathsf{OT},\mathcal{S}}^2$ in a successful execution.
9. To generate the final round message,
   (a) If $f_0 = f_1$, then $\mathsf{Sim}_{\mathcal{S}}^2$ generates the final round message for $b = 0$ as shown below and sends the same final round message for $b = 1$.
   (b) $\mathsf{Sim}_{\mathcal{S}}^2$ recovers $\{x_{i,j}\}_{i\in[m],j\in[\ell]}$ from x_{OT}.
   (c) $\mathsf{Sim}_{\mathcal{S}}^2$ runs Sim_Φ by giving $\{x_i\}_{i\notin K}$ as the first round message sent by the malicious client and (f_0, f_1) as the function to be computed. When Sim_Φ makes a query to the ideal functionality on input $x_{\mathcal{R}}$, $\mathsf{Sim}_{\mathcal{S}}^2$ intercepts this query and makes a query on $(x_{\mathcal{R}}, f_0, f_1)$ to its own functionality and obtains $\mathsf{out}_0, \mathsf{out}_1$. It then provides $(\mathsf{out}_0, \mathsf{out}_1)$ as the output from the ideal functionality to Sim_Φ and obtains $\{z_i[b]\}_{i\notin K,b\in\{0,1\}}$.
   (d) It computes $\widetilde{C}, \{\mathsf{lab}^i\}_{i\in[m]} \leftarrow \mathsf{Sim}_{\mathsf{GC}}(1^\lambda, 1^{|C|}, 1^m, \{s_i\}_{i\in K})$ (where $\{s_i\}_{i\in K}$ is recovered from the random tape of $\mathcal{A}_2$).
   (e) For each $i \in [m]$ and $b \in \{0,1\}$,
      i. $\mathsf{Sim}_{\mathcal{S}}^2$ computes $\mathsf{Com}_3^i[b]$ honestly using the input and randomness used in generating Com_1^i (obtained from the random tape of $\mathcal{A}_2$).
      ii. It computes $ct_0^i := \mathsf{Enc}(sk_0^i, \mathsf{lab}^i)$ and $ct_1^i := \mathsf{Enc}(sk_1^i, \mathsf{lab}^i)$.
      iii. If $i \in K$, then
         - It computes $r_i[b] := \mathsf{PRF}_{\mathsf{key}_i}(f_b)$.
         - It computes $\widetilde{\Phi}_{2,i}[b], \{\overline{\mathsf{lab}}_0^{i,j}[b], \overline{\mathsf{lab}}_1^{i,j}[b]\}_{j\in[\ell]}, \{\mathsf{lab}_0^{i,j,k}[b], \mathsf{lab}_1^{i,j,k}[b]\}_{j\in[\ell],k\in[\lambda]} \leftarrow \mathsf{Garble}(1^\lambda, \Phi_{2,i}[b]; r_i[b])$.
      iv. If $i \notin K$, then
         - It computes $\widetilde{\Phi}_{2,i}[b], \{\overline{\mathsf{lab}}^{i,j}[b]\}_{j\in[\ell]}, \{\mathsf{lab}^{i,j,k}[b]\}_{j\in[\ell],k\in[\lambda]} \leftarrow \mathsf{Sim}_{\mathsf{GC}}(1^\lambda, 1^{|\Phi_{2,i}[b]|}, 1^{2\ell}, z_i[b])$.
         - For each $j \in [\ell]$ and $k \in [\lambda]$, it sets $\mathsf{lab}_0^{i,j,k}[b] = \mathsf{lab}_1^{i,j,k}[b] = \mathsf{lab}^{i,j,k}[b]$.
      v. For each $j \in [\ell]$ and $k \in [\lambda]$, it computes $ct_0^{i,j,k}[b] := \mathsf{Enc}(sk_0^{i,j,k}, \mathsf{lab}_0^{i,j,k}[b])$ and $ct_1^{i,j,k}[b] := \mathsf{Enc}(sk_1^{i,j,k}, \mathsf{lab}_1^{i,j,k}[b])$.

(f) It finally outputs $\widetilde{C}, \{ct_0^i, ct_1^i, \mathsf{Com}_3^i[b], \mathsf{ots}_4^i[b], \widetilde{\Phi}_{2,i}[b]\}_{i\in[m],b\in\{0,1\}}$ and $\{\mathsf{otm}^{i,j,k}[b]_4, \overline{\mathsf{lab}}^{i,j}[b], ct^{i,j,k}[b]_0, ct^{i,j,k}[b]_1\}_{i\in[m],j\in[\ell],k\in[\lambda],b\in\{0,1\}}$ as the final round message.

(g) $\mathsf{Sim}_{\mathcal{S}}^2$ has an internal step counter and if it runs for more than 2^λ time, it aborts and outputs a special symbol $\mathsf{time\text{-}out}$.

Running time of $\mathsf{Sim}_{\mathcal{S}}^1$ and $\mathsf{Sim}_{\mathcal{S}}^2$. Let ϵ be the probability that $\mathsf{Sim}_{\mathsf{OT},\mathcal{S}}^1$ outputs $x_{\mathsf{OT}} \neq \mathsf{abort}$ in the description of $\mathsf{Sim}_{\mathcal{S}}^1$. Then the expected running time of $\mathsf{Sim}_{\mathcal{S}}^1$ is given by $\mathsf{poly}(\lambda)+\mathsf{poly}(\lambda)\epsilon(12\lambda/\epsilon) = \mathsf{poly}(\lambda)$. Via a standard argument (see [GK96a]), we can show that $\widetilde{\epsilon}$ is within a factor of 2 of the value of ϵ except with probability $2^{-\lambda}$. Thus, the expected running time of $\mathsf{Sim}_{\mathcal{S}}^2$ is given by $\mathsf{poly}(\lambda) + \mathsf{poly}(\lambda)\epsilon(O(\lambda^2/\epsilon) + 2^\lambda(1/2^\lambda)) = \mathsf{poly}(\lambda)$.

Proof of Indistinguishability. We now show that the real experiment and the ideal experiment involving $(\mathsf{Sim}_{\mathcal{S}}^1, \mathsf{Sim}_{\mathcal{S}}^2)$ are computationally indistinguishable via a hybrid argument.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the output of the real experiment (i.e., Expt_1) in Figure 4.
- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we make the following changes. We define an adversary $\mathcal{A}'$ that interacts with $\mathcal{A}$ internally and receives the messages corresponding to the OT protocol externally. Specifically, $\mathcal{A}'$ does the following:
  - $\mathcal{A}'$ initializes $\mathcal{A}$ and obtains the first round messages $\{\mathsf{otm}_1^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{\mathsf{otm}_1^i\}_{i\in[m]}$ from $\mathcal{A}$. It forwards these messages externally.
  - $\mathcal{A}'$ receives $\{\mathsf{otm}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{\mathsf{otm}_2^i\}_{i\in[m]}$ externally.
  - For each $i \in [m]$, $\mathcal{A}_1$ generates $\mathsf{Com}_1^i \leftarrow \mathsf{ECom}_1(1^\lambda, (y_i, \mathsf{key}_i, (sk_0^{i,j}, sk_1^{i,j})); s_i)$ (for uniformly chosen s_i).
  - $\mathcal{A}'$ now runs $\mathcal{A}$ on $\{\mathsf{otm}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{\mathsf{Com}_1^i, \mathsf{otm}_2^i\}_{i\in[m]}$.
  - $\mathcal{A}'$ receives $\{f_b\}_{b\in\{0,1\}}, \{\mathsf{otm}_3^{i,j,k}[b]\}_{i\in[m],j\in[\ell],k\in[\lambda],b\in\{0,1\}}, \{\mathsf{Com}_2^i[b], \mathsf{otm}_3^i[b]\}_{i\in[m],b\in\{0,1\}}$ from $\mathcal{A}$ and forwards $\{\mathsf{otm}_3^{i,j,k}[b]\}_{i\in[m],j\in[\ell],k\in[\lambda],b\in\{0,1\}}, \{\mathsf{otm}_3^i[b]\}_{i\in[m],b\in\{0,1\}}$ externally.

  We run $\mathsf{Sim}_{\mathsf{OT},\mathcal{S}}^1$ and $\mathsf{Sim}_{\mathsf{OT},\mathcal{S}}^2$ on $\mathcal{A}'$ and obtain the view of $\mathcal{A}'$ as well the messages in the OT protocol. We generate the final round message to $\mathcal{A}$ using the internal randomness of $\mathcal{A}'$.

  In Claim C.3, we show that Hyb_0 and Hyb_1 are computationally indistinguishable from the 1-rewinding sender security property of the OT protocol.
- $\underline{\mathsf{Hyb}_2}$: In this hybrid,
  - We estimate $\widetilde{\epsilon}$ as in the simulation using $\mathcal{A}'$. We output the special symbol $\mathsf{timeout}$, if we run for 2^λ steps.
  - We run $\mathsf{Sim}_{\mathsf{OT},\mathcal{S}}^2$ on $\mathcal{A}'$ for $\lambda^2/\widetilde{\epsilon}$ independent executions and we output the special symbol fail, if $\mathsf{Sim}_{\mathsf{OT},\mathcal{S}}^2$ outputs fail, in each of these executions. We output the special symbol $\mathsf{timeout}$, if we run for 2^λ steps.

  In Claim C.4, we show that $\mathsf{Hyb}_1 \approx_s \mathsf{Hyb}_2$.
- $\underline{\mathsf{Hyb}_3}$: In this hybrid, we run $\mathsf{Sim}_{\mathsf{OT},\mathcal{S}}^1$ on the adversary $\mathcal{A}_1$ (defined in the simulation), instead of running it on $\mathcal{A}'$ as in the previous hybrid.

  We show in Claim C.5 that $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$ from the 1-rewinding security of the extractable commitment scheme.

- $\underline{\mathsf{Hyb}_4}$: In this hybrid, we compute $ct^i_{1-K_i}$ as $\mathsf{Enc}(sk^i_{1-K_i}, \mathsf{lab}^i_{K_i})$ instead of $\mathsf{Enc}(sk^i_{1-K_i}, \mathsf{lab}^i_{1-K_i})$ for each $i \in [m]$. It now follows from the semantic security of the encryption scheme that $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$.

- $\underline{\mathsf{Hyb}_5}$: In this hybrid, we generate $\widetilde{C}, \{\mathsf{lab}^i_{K_i}\}_{i\in[m]}$ using $\mathsf{Sim}_{\mathsf{GC}}(1^\lambda, 1^{|C|}, 1^m, \{s_i\}_{i\in K})$ instead of using the Garble procedure. It follows from the security of the garbling scheme that $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$.

- $\underline{\mathsf{Hyb}_6}$: In this hybrid, we run $\mathsf{Sim}^2_{\mathsf{OT},\mathcal{S}}$ on $\mathcal{A}_2$ instead of $\mathcal{A}'$.

  We show in Claim C.6 that $\mathsf{Hyb}_5 \approx_c \mathsf{Hyb}_6$ from the 1-rewinding security of the extractable commitment scheme.

- $\underline{\mathsf{Hyb}_7}$: In this hybrid, for each $i \in [m]$, $j \in [\ell]$, $k \in [\lambda]$, and $b \in \{0,1\}$, we generate the randomness used in computing $ct^{i,j,k}_0[b]$ as the output of truly random function on input f_b instead of using a PRF.

  It follows directly from the security of the PRF that $\mathsf{Hyb}_6 \approx_c \mathsf{Hyb}_7$.

- $\underline{\mathsf{Hyb}_8}$: In this hybrid, for each $i \notin K$ and $b \in \{0,1\}$, we compute $r_i[b]$ (used as randomness in computing $\widetilde{\Phi}_{2,i}$) as the output of a random function on the input f_b rather than using a PRF.

  As in the previous hybrid, it follows directly from the security of the PRF that $\mathsf{Hyb}_7 \approx_c \mathsf{Hyb}_8$.

- $\underline{\mathsf{Hyb}_9}$: In this hybrid, for each $i \notin K$, $j \in [\ell]$, $k \in [\lambda]$ and $b \in \{0,1\}$, we compute $ct^{i,j,k}_{1-x_{i,j,k}}[b] \leftarrow \mathsf{Enc}(sk^{i,j,k}_{1-x_{i,j,k}}, \mathsf{lab}^{i,j,k}_{x_{i,j,k}}[b])$.

  In Claim C.7, we show that $\mathsf{Hyb}_8 \approx_c \mathsf{Hyb}_9$ from the semantic security of the symmetric key encryption.

- $\underline{\mathsf{Hyb}_{10}}$: In this hybrid, for each $i \notin K$ and $b \in \{0,1\}$, we generate $\widetilde{\Phi}_{2,i}[b], \{\overline{\mathsf{lab}}^{i,j}_{y_{i,j}}[b]\}_{j\in[\ell]}$ and $\{\mathsf{lab}^{i,j,k}_{x_{i,j,k}}[b]\}_{j\in[\ell],k\in[\lambda]}$ as the output of $\mathsf{Sim}_{\mathsf{GC}}$ rather than using Garble procedure.

  It follows from the security of garbling scheme that $\mathsf{Hyb}_9 \approx_c \mathsf{Hyb}_{10}$.

- $\underline{\mathsf{Hyb}_{11}}$: In this hybrid, we use the simulator Sim_Φ to generate the first round messages $\{y_i\}_{i\in K}$ from honest sender as well as the second round messages $\{z_i[b]\}_{i\notin K, b\in\{0,1\}}$ from the honest servers.

  In Claim C.8, we show that $\mathsf{Hyb}_{10} \approx_s \mathsf{Hyb}_{11}$ from the security of the outer protocol. Note that Hyb_{11} is identical to the output of the ideal experiment (i.e., Expt_2 in Figure 4).

Claim C.3. *Assuming the 1-rewinding sender security of the oblivious transfer, we have* $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$.

Proof. Assume for the sake of contradiction that there exists a distinguisher D that can distinguish between Hyb_0 and Hyb_1 with non-negligible advantage. We will use D to construct an adversary $\mathcal{B}$ that breaks the 1-rewinding sender security of the OT protocol.

- For each $i \in [m]$, $\mathcal{B}$ samples sk^i_0, sk^i_1 from $\mathsf{KeyGen}(1^\lambda)$ and for each $j \in [\ell]$ and $k \in [\lambda]$, it samples $sk^{i,j,k}_0, sk^{i,j,k}_1$ from $\mathsf{KeyGen}(1^\lambda)$. It sends $\{sk^i_0, sk^i_1\}_{i\in[m]}$ and $\{sk^{i,j,k}_0, sk^{i,j,k}_1\}_{i\in[m],j\in[\ell]}$ as the challenge sender inputs to the external challenger.

- $\mathcal{B}$ constructs $\mathcal{A}'$ as in the description of Hyb_1 and forwards the messages from $\mathcal{A}'$ to external challenger and vice-versa.

- The external challenger finally outputs the view of $\mathcal{A}'$. $\mathcal{B}$ uses this to generate the final round message of the protocol Π and generate the view of $\mathcal{A}$.
- $\mathcal{B}$ finally runs D on the above generated view of $\mathcal{A}$ and outputs whatever D outputs.

Note that if the view of $\mathcal{A}'$ is generated using the real algorithms then the inputs to D are identically distributed to Hyb_0. Otherwise, they are distributed identically to Hyb_1. Thus, if D can distinguish between Hyb_0 and Hyb_1 with non-negligible advantage, then $\mathcal{A}$ breaks the 1-rewinding sender security property of the OT protocol and this is a contradiction. □

Claim C.4. $\mathsf{Hyb}_1 \approx_s \mathsf{Hyb}_2$.

Proof. Via a standard argument (see [GK96a]) that, $\widetilde{\epsilon}$ is within a factor of two of ϵ except with probability $2^{-\lambda}$. Therefore, the expected running time to generate the output of Hyb_2 is $\mathsf{poly}(\lambda) + \mathsf{poly}(\lambda)\epsilon(12\lambda/\epsilon + 1/2^{\lambda} * 2^{\lambda} + O(\lambda^2/\epsilon)) = \mathsf{poly}(\lambda)$. Thus, from Markov's inequality, it follows that the probability that we output the special symbol $\mathsf{timeout}$ in Hyb_2 is negligible. Furthermore, it follows from claim C.3 that the probability that $\mathsf{Sim}^2_{\mathsf{OT},\mathcal{S}}$ outputs the special symbol fail in any execution is negligible. Thus, we output the special symbol fail in Hyb_2 is negligible. □

Claim C.5. *Assuming the 1-rewinding security of the extractable commitment protocol, we have* $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$.

Proof. Assume for the sake of contradiction that there exists a distinguisher D that can distinguish between Hyb_2 and Hyb_3 with non-negligible advantage. We will use D to construct an adversary $\mathcal{B}$ that breaks the 1-rewinding security of the extractable commitment scheme.

- For each $i \in [m]$, $\mathcal{B}$ sends $(y_i, \mathsf{key}_i, \{sk_0^{i,j,k}, sk_1^{i,j,k}\}_{j\in[\ell],k\in[\lambda]})$ and $\perp$ as the challenge messages to the external challenger.
- Whenever $\mathsf{Sim}^1_{\mathsf{OT},\mathcal{S}}$ sends a fresh second round OT message $\{\mathsf{otm}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{\mathsf{otm}_2^i\}_{i\in[m]}$, $\mathcal{B}$ queries the external challenger and receives $\{\mathsf{Com}_1^i\}_{i\in[m]}$. $\mathcal{B}$ runs $\mathcal{A}$ on $\{\mathsf{otm}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$, $\{\mathsf{Com}_1^i, \mathsf{otm}_2^i\}_{i\in[m]}$ and forwards the third round OT messages to $\mathsf{Sim}^1_{\mathsf{OT},\mathcal{S}}$.
- $\mathcal{B}$ generates the rest of the protocol messages in Hyb_2 and finally generates the view of the adversary $\mathcal{A}$ at the end of the execution.
- $\mathcal{B}$ finally runs D on the view of $\mathcal{A}$ and outputs whatever D outputs.

Note that if the commitments generated by the external challenger contain the message $\perp$, then the inputs to D are distributed identically to Hyb_3. Otherwise, it is distributed identically to Hyb_2. Thus, if $\mathcal{B}$ can distinguish between Hyb_2 and Hyb_3 with non-negligible advantage, then $\mathcal{B}$ can break the 1-rewinding security property of the extractable commitment scheme which is a contradiction. □

Claim C.6. *Assuming the 1-rewinding security of the extractable commitment protocol, we have* $\mathsf{Hyb}_5 \approx_c \mathsf{Hyb}_6$.

Proof. Assume for the sake of contradiction that there exists a distinguisher D that can distinguish between Hyb_5 and Hyb_6 with non-negligible advantage. We will use D to construct an adversary $\mathcal{B}$ that breaks the 1-rewinding security of the extractable commitment scheme.

- For each $i \notin K$, $\mathcal{B}$ sends $(y_i, \mathsf{key}_i, \{sk_0^{i,j}, sk_1^{i,j}\}_{j\in[\ell]})$ and $\perp$ as the challenge messages to the external challenger.
- Whenever $\mathsf{Sim}^2_{\mathsf{OT},\mathcal{S}}$ sends a fresh second round OT message $\{\mathsf{otm}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{\mathsf{otm}_2^i\}_{i\in[m]}$, $\mathcal{B}$ queries the external challenger and receives $\{\mathsf{Com}_1^i\}_{i\notin K}$. $\mathcal{B}$ generates $\{\mathsf{Com}_1^i\}_{i\in K}$ as in Hyb_5.
- $\mathcal{B}$ runs $\mathcal{A}$ on $\{\mathsf{otm}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{\mathsf{Com}_1^i, \mathsf{otm}_2^i\}_{i\in[m]}$ and obtains forwards the third round OT messages to $\mathsf{Sim}^2_{\mathsf{OT},\mathcal{S}}$.
- When $\mathsf{Sim}^2_{\mathsf{OT},\mathcal{S}}$ outputs the third round and the fourth round messages of the OT protocol, $\mathcal{B}$ recovers the corresponding $\{\mathsf{Com}_2^i[b]\}_{i\notin K}$ from its interaction with $\mathcal{A}$ and it sends $\{\mathsf{Com}_2^i[b]\}_{i\notin K}$ to the external challenger and obtains $\{\mathsf{Com}_3^i[b]\}_{i\notin K}$.
- $\mathcal{B}$ generates the rest of the protocol messages in Hyb_5 and finally generates the view of the adversary $\mathcal{A}$ at the end of the execution.
- $\mathcal{B}$ finally runs D on the view of $\mathcal{A}$ and outputs whatever D outputs.

Note that if the commitments generated by the external challenger contain the message $\perp$, then the inputs to D are distributed identically to Hyb_6. Otherwise, it is distributed identically to Hyb_5. Thus, if $\mathcal{B}$ can distinguish between Hyb_5 and Hyb_6 with non-negligible advantage, then $\mathcal{B}$ can break the 1-rewinding security property of the extractable commitment scheme which is a contradiction. □

Claim C.7. *Assuming the semantic security of the encryption scheme, we have* $\mathsf{Hyb}_8 \approx_c \mathsf{Hyb}_9$.

Proof. Assume for the sake of contradiction that there exists a distinguisher D that can distinguish between Hyb_8 and Hyb_9 with non-negligible advantage. We will use D to construct an adversary $\mathcal{B}$ that breaks the semantic security of the encryption scheme.

- For each $i \notin K$, $j \in [\ell]$, $k \in [\lambda]$ and $b \in \{0,1\}$, $\mathcal{A}$ provides $\mathsf{lab}^{i,j,k}_{x_{i,j,k}}[b]$ as the left challenge message and $\mathsf{lab}^{i,j,k}_{1-x_{i,j,k}}[b]$ as the right challenge message. It receives $ct^{i,j,k}_{1-x_{i,j,k}}[b]$ from the challenger.
- It generates the rest of the protocol messages as in Hyb_8. Note that this does not require knowledge of $sk^{i,j,k}_{1-x_{i,j,k}}$.
- It generates the view of the adversary $\mathcal{A}$ and finally runs D on this view and outputs whatever D outputs.

Note that if the challenge ciphertexts correspond to the encryption of the left message, then the inputs to D are identical to Hyb_9. Otherwise, the inputs are identical to Hyb_8. Thus, if D can distinguish between Hyb_8 and Hyb_9 with non-negligible advantage, then $\mathcal{B}$ can break the semantic security of the encryption scheme and this is a contradiction. □

Claim C.8. *Assuming the security of the outer MPC protocol Φ, we have that* $\mathsf{Hyb}_{10} \approx_s \mathsf{Hyb}_{11}$.

Proof. Assume for the sake of contradiction that there exists a distinguisher D that can distinguish between Hyb_{10} and Hyb_{11} with non-negligible advantage. We will use D to construct an adversary $\mathcal{B}$ that breaks the security of the protocol Φ.

- $\mathcal{B}$ corrupts the receiver client and sends $x_{\mathcal{S}}$ as the input of the sender client to the external challenger.
- $\mathcal{B}$ runs $\mathsf{Sim}^1_{\mathsf{OT},\mathcal{S}}$ on $\mathcal{A}_1$ and obtains $\{K_i\}_{i\in[m]}$ and $\{x_{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$. Let K be the set of indices i such that $K_i = 1$. If $|K| > \lambda$, it resets $K = \emptyset$.
- $\mathcal{B}$ asks the challenger to corrupt the set of servers corresponding to the set K and obtains $\{y_i\}_{i\in K}$.
- $\mathcal{B}$ sends $\{x_i\}_{i\notin K}$ as the first round message from the malicious receiver to the honest servers.
- At the end of execution of $\mathsf{Sim}^2_{\mathsf{OT},\mathcal{S}}$, $\mathcal{B}$ recovers (f_0, f_1) from the third round message, and sends this to the external challenger.
- The challenger replies with $\{z_i[b]\}_{i\notin K,b\in\{0,1\}}$. It uses this information to compute $\{\widetilde{\Phi}_{2,i}[b]\}_{i\notin K,b\in\{0,1\}}$.
- It generates the view of the adversary and then runs D on this view.

Note that if the outer protocol messages given by the challenger are generated using the real algorithms then the inputs to D are identical to Hyb_{10}. Otherwise, they are identical to Hyb_{11}. Thus, if if D can distinguish between Hyb_{10} and Hyb_{11} with non-negligible advantage, then $\mathcal{B}$ can break the security of the outer protocol Φ and this is a contradiction. $\square$

C.2.2 Receiver Security

We begin with the descriptions of the simulators $(\mathsf{Sim}^1_{\mathcal{R}}, \mathsf{Sim}^2_{\mathcal{R}})$.

Description of $\mathsf{Sim}^1_{\mathcal{R}}$.

1. $\mathsf{Sim}^1_{\mathcal{R}}$ constructs an adversary $\mathcal{A}_1$ that interacts with $\mathcal{A}$ internally and obtains the messages of the OT protocol externally. Specifically, $\mathcal{A}_1$ does the following:
    - It receives $\{\mathsf{otm}_1^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{\mathsf{otm}_1^i\}_{i\in[m]}$ externally and sends this to $\mathcal{A}$.
    - It receives $\{\mathsf{otm}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{\mathsf{Com}_1^i, \mathsf{otm}_2^i\}_{i\in[m]}$ from $\mathcal{A}$ and sends $\{\mathsf{otm}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$, $\{\mathsf{otm}_2^i\}_{i\in[m]}$ externally.
    - On receiving $\{\mathsf{otm}_3^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{\mathsf{otm}_3^i\}_{i\in[m]}$ externally, $\mathcal{A}_1$ samples Com_i^2 using $\mathsf{ECom}_2(\mathsf{Com}_1^i)$ for each $i \in [m]$ and runs $\mathcal{A}$ on f, $\{\mathsf{otm}_3^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{\mathsf{Com}_2^i, \mathsf{otm}_3^i\}_{i\in[m]}$.
    - It receives the last round message from $\mathcal{A}$ and checks if for each $i \in [m]$, whether Com_3^i is valid. If yes, it forwards $\{\mathsf{otm}_4^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{\mathsf{otm}_4^i\}_{i\in[m]}$ externally and otherwise, it aborts.
2. $\mathsf{Sim}^1_{\mathcal{R}}$ runs $\mathsf{Sim}^1_{\mathsf{OT},\mathcal{R}}$ on $\mathcal{A}_1$ and obtains $\{\mathsf{otm}_1^{i,j,k}, \mathsf{otm}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$, $\{\mathsf{otm}_1^i, \mathsf{otm}_2^i\}_{i\in[m]}$ along with $\{sk_0^{i,j,k}, sk_1^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ and $\{sk_0^i, sk_1^i\}_{i\in[m]}$. When $\mathsf{Sim}^1_{\mathsf{OT},\mathcal{R}}$ is interacting with $\mathcal{A}_1$, $\mathsf{Sim}^1_{\mathcal{R}}$ observes this interaction along with the internal interaction of $\mathcal{A}_1$ with $\mathcal{A}$. Whenever $\mathcal{A}_1$ forwards the final round message externally, it means that $\{\mathsf{Com}_3^i\}_{i\in[m]}$ is valid. $\mathsf{Sim}^1_{\mathcal{R}}$ uses this to extract $\{y_i, \mathsf{key}_i, \{sk_0'^{i,j,k}, sk_1'^{i,j,k}\}_{j\in[\ell],k\in[\lambda]}\}_{i\in[m]}$.
3. $\mathsf{Sim}^1_{\mathcal{R}}$ runs Sim_Φ on input $y_1, \ldots, y_m$ and function f. When Sim_Φ queries the ideal functionality on $x_{\mathcal{S}}$, $\mathsf{Sim}^1_{\mathcal{R}}$ intercepts this query and records it.

4. $\mathsf{Sim}^1_{\mathcal{R}}$ outputs $x_{\mathcal{S}}$, the first and second round messages of the protocol Π and $sk = (sk_{\mathsf{OT}}, \{y_i, \mathsf{key}_i, \{sk_0'^{i,j,k}, sk_1'^{i,j,k}\}_{j\in[\ell],k\in[\lambda]}\}_{i\in[m]}, \{sk_0^{i,j,k}, sk_1^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, \{sk_0^i, sk_1^i\}_{i\in[m]},$ random tape of $\mathcal{A}_1$ and Sim_Φ.).

Description of $\mathsf{Sim}^2_{\mathcal{R}}$.

1. $\mathsf{Sim}^2_{\mathcal{R}}$ runs $\mathsf{Sim}^2_{\mathsf{OT},\mathcal{R}}$ on $\mathcal{A}_1$ with input sk_{OT} and obtains $\{\mathsf{otm}_3^{i,j,k}, \mathsf{otm}_4^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ and $\{\mathsf{otm}_3^i, \mathsf{otm}_4^i\}_{i\in[m]}$. It also obtains $\{\mathsf{Com}_2^i\}_{i\in[m]}$ from the random tape of $\mathcal{A}_1$. Finally, it recovers the rest of the final round message that $\mathcal{A}$ has sent to $\mathcal{A}_1$.

2. For each $i \in [m]$, $\mathsf{Sim}^2_{\mathcal{R}}$ does the following:

   (a) For each $j \in [\ell]$ and $k \in [\lambda]$, it computes $\mathsf{lab}_b^{i,j,k} := \mathsf{Dec}(sk_b^{i,j,k}, ct_b^{i,j,k})$ for each $b \in \{0,1\}$.

   (b) It computes $\mathsf{lab}_b^i := \mathsf{Dec}(sk_b^i, ct_b^i)$ for each $b \in \{0,1\}$.

   (c) It computes $r_i := \mathsf{PRF}_{\mathsf{key}_i}(f)$.

   (d) It computes $\widetilde{\Phi}_{2,i}, \{\widetilde{\mathsf{lab}}_0^{i,j}, \widetilde{\mathsf{lab}}_1^{i,j}\}_{j\in[\ell]}, \{\mathsf{lab}_0'^{i,j,k}, \mathsf{lab}_1'^{i,j,k}\}_{j\in[\ell]} \leftarrow \mathsf{Garble}(1^\lambda, \Phi_{2,i}; r_i)$.

   (e) It checks if $\widetilde{\Phi}_{2,i}$ that is received in the fourth round is the same as the one computed above.

   (f) For each $j \in [\ell]$ and $k \in [\lambda]$, it checks if $sk_b^{i,j,k} = sk_b'^{i,j,k}$ for each $b \in \{0,1\}$. It then checks if $\mathsf{lab}_b^{i,j,k} = \mathsf{lab}_b'^{i,j,k}$ for each $b \in \{0,1\}$.

   (g) For each $j \in [\ell]$, it checks if $\widetilde{\mathsf{lab}}_{y_{i,j}}^{i,j}$ is same as the label obtained from $\mathcal{A}$ in the last round.

   (h) If any of the checks fail, it adds i to a set C (which is initially empty).

3. If $|C| > \lambda$, then $\mathsf{Sim}^2_{\mathcal{R}}$ sends an abort to the ideal functionality.

4. If $|C| \le \lambda$ then $\mathsf{Sim}^2_{\mathcal{R}}$ continues the execution of Sim_Φ and instructs Sim_Φ to adaptively corrupt the servers indexed by the set C and obtains $\{x_i\}_{i\in C}$.

5. It then chooses a random subset K of size λ and computes $\{s_i\}_{i\in K} \leftarrow \mathsf{Eval}(\widetilde{C}, \{\mathsf{lab}^i_{K_i}\}_{i\in[m]})$.

6. It then instructs Sim_Φ to adaptively corrupt the set of servers indexed by K and obtains $\{x_i\}_{i\in K}$. It then does the exact same checks as the honest party in output computing phase. If any of the checks fail, then $\mathsf{Sim}^2_{\mathcal{R}}$ sends abort to the ideal functionality.

7. For each $i \in C \cup K$, $\mathsf{Sim}^2_{\mathcal{R}}$ computes $z_i := \mathsf{Eval}(\widetilde{\Phi}_{2,i}, \{\overline{\mathsf{lab}}^{i,j}_{y_{i,j}}\}_{j\in[\ell]}, \{\mathsf{lab}^{i,j,k}_{x_{i,j,k}}\}_{j\in[\ell],k\in[\lambda]})$.

8. It then sends $\{z_i\}_{i\in C\cup K}$ as the second round message from the corrupted servers to Sim_Φ. If Sim_Φ instructs the honest client to abort, then $\mathsf{Sim}^2_{\mathcal{S}}$ sends abort to the ideal functionality. Otherwise, it asks the ideal functionality to deliver outputs to the honest receiver.

Proof of Indistinguishability. We now show that the real experiment and the ideal experiment are computationally indistinguishable using a hybrid argument.

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

- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we generate the OT protocol messages by running $(\mathsf{Sim}^1_{\mathsf{OT},\mathcal{R}}, \mathsf{Sim}^2_{\mathsf{OT},\mathcal{R}})$ on the adversary $\mathcal{A}_1$ (described in the simulation).

  We show that the Hyb_0 is computationally indistinguishable from Hyb_1 from the receiver security of the OT protocol in Claim C.9.

- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we extract $(y_i, \mathsf{key}_i, \{sk_0^{i,j,k}, sk_1^{i,j,k}\}_{j\in[\ell],k\in[\lambda]})$ for each $i \in [m]$ from the extractable commitment scheme as explained in the simulation.

  This hybrid is identical to the previous hybrid as it does not affect the view of the adversary or the outputs of the honest parties.

- $\underline{\mathsf{Hyb}_3}$: In this hybrid, we construct the set C as explained in the simulation and if $|C| > \lambda$, we instruct the honest party to abort.

  In Claim C.10, we show 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 messages $\{x_i\}_{i\in C\cup K}$ and the output of the honest receiver. Note that Hyb_4 is identical to the output of the ideal experiment.

  In Claim C.11, we show that $\mathsf{Hyb}_3 \approx_s \mathsf{Hyb}_4$ from the security of the outer MPC protocol.

Claim C.9. *Assuming the receiver security property of the oblivious transfer, we have* $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$.

Proof. Assume for the sake of contradiction that there exists a distinguisher D that can distinguish between Hyb_0 and Hyb_1 with non-negligible advantage. We will use D to construct an adversary $\mathcal{B}$ that breaks the receiver security of the OT protocol.

- $\mathcal{B}$ provides $\{x_{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ and $\{K_i\}_{i\in[m]}$ as the receiver inputs to the external challenger.
- It constructs the adversary $\mathcal{A}_1$ as described in the simulation. It interacts with the external challenger and forwards the messages from it to $\mathcal{A}_1$. The messages sent by $\mathcal{A}_1$ are conveyed back to the challenger.
- Finally, $\mathcal{B}$ obtains the view of $\mathcal{A}'$ and the output of the honest receiver in the OT protocol. It uses the view of $\mathcal{A}'$ to generate the view of $\mathcal{A}$. It then uses the output of the honest receiver in the OT protocol to do the exact same checks as in Hyb_0 and computes the output of the honest receiver in protocol Π.
- $\mathcal{B}$ finally runs D on the view of $\mathcal{A}$ and the output of the honest receiver in Π and outputs whatever D outputs.

Note that if the OT protocol messages are generated by the challenger using the real algorithms, then the inputs to D are distributed identically to Hyb_0. Otherwise, they are distributed identically to Hyb_1. Thus, if D can distinguish between Hyb_0 and Hyb_1 with non-negligible advantage, then $\mathcal{B}$ can break the receiver security of the OT protocol and this is a contradiction. □

Claim C.10. $\mathsf{Hyb}_2 \approx_s \mathsf{Hyb}_3$.

Proof. We first argue that for each $i \in C \cap K$, one of the checks done by the honest receiver in Hyb_2 independently fails with probability at least 1/2. To see why this is the case, consider some $i \in K \cap C$. If i was added to C as a result of the check in Step 2.(e) failing, then the honest receiver also catches this with probability 1. If i was added to C as a result of a check in Step 2.(f) fails, then it means that there exists j, k and $b \in \{0,1\}$ such that $sk_b'^{i,j,k} \neq sk_b^{i,j,k}$. In this case, the honest receiver catches this with probability at least 1/2 since $x_{i,j,k} = b$ with probability 1/2. If i was added to C as a result of the check in Step 2.(g) failing, then honest receiver in Hyb_2 catches this with probability 1. This shows that for each $i \in C \cap K$, one of the checks done by the honest receiver in Hyb_2 independently fails with probability at least 1/2.

To complete the proof of the claim, we now show that if K is chosen uniformly at random then the probability that $|K \cap C| \leq \lambda/100$ is $2^{-O(\lambda)}$. Consider an arbitrary subset $C' \subseteq C$ of size $\lambda/100$. We first upper bound the probability that $K \cap C \subseteq C'$.

$$\begin{aligned}
\Pr[K \cap C \subseteq C'] &= \frac{\binom{m - |C \setminus C'|}{\lambda}}{\binom{m}{\lambda}} \\
&< \frac{\binom{m - 99\lambda/100}{\lambda}}{\binom{m}{\lambda}} \\
&< \frac{(m - 99\lambda/100)!}{\lambda!(m - 199\lambda/100)!} \cdot \frac{\lambda!(m-\lambda)!}{m!} \\
&= \frac{(m - 99\lambda/100)!(m-\lambda)!}{m! \cdot (m - 199\lambda/100)!} \\
&< (1 - 99\lambda/100m)^{\lambda} \\
&< e^{-99\lambda/800}.
\end{aligned}$$

The total number of subsets of C of size $\lambda/100$ is at most $\binom{m}{\lambda/100} < (100em/\lambda)^{\lambda/100} = (800e)^{\lambda/100} < e^{8\lambda/100}$ (since $800e < e^8$). Thus, via a standard union bound, the probability that there exists a subset C' of C of size $\lambda/100$ such that $|K \cap C| \subset C'$ is upper bounded by $e^{-O(\lambda)}$. This completes the proof of the claim. $\square$

Claim C.11. *Assuming the security of the outer MPC protocol, we have* $\mathsf{Hyb}_3 \approx_s \mathsf{Hyb}_4$.

Proof. 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 will use D to construct an adversary $\mathcal{B}$ that breaks the security of the protocol Φ.

- $\mathcal{B}$ corrupts the sender client and sends $x_{\mathcal{R}}$ as the input of the receiver client to the external challenger.
- $\mathcal{B}$ extracts $(y_1, \ldots, y_m)$ as the first round message from the corrupted client to the servers.
- Before receiving the second round message from the challenger, it asks the challenger to adaptively corrupt the servers indexed by the set C and the set K.
- $\mathcal{B}$ receives $\{x_i\}_{i \in C \cup K}$.
- It uses this to perform all the checks as in Hyb_3 and evaluates the garbled circuits $\widetilde{\Phi}_{2,i}$ for each $i \in C \cup K$ and obtains $\{z_i\}_{i \in C \cup K}$.
- It sends $\{z_i\}_{i \in C \cup K}$ to the challenger and obtains the output of the honest receiver.
- It computes the view of the adversary $\mathcal{A}$ and then runs D on this view and the output of the honest receiver.

Note that if the outer protocol messages given by the challenger are generated using the real algorithms then the inputs to D are identical to Hyb_3. Otherwise, they are identical to Hyb_4. Thus, if if D can distinguish between Hyb_3 and Hyb_4 with non-negligible advantage, then $\mathcal{B}$ can break the security of the outer protocol Φ and this is a contradiction. □

D Rewind Secure Oblivious Transfer

In this section, we give a construction of 1-rewinding sender secure oblivious transfer protocol needed to instantiate the 2-party computation protocol in Section C. Our construction is based on the protocol given in [ORS15] and makes black-box access to a public key encryption with pseudorandom public keys. The main theorem we show is:

Theorem D.1. *Assume black-box access to a public-key encryption with pseudorandom public keys. There exists a four round, 1-out-of-2 parallel-secure OT protocol with 1-rewinding sender security.*

We first show a protocol that is standalone secure and then parallel security follows immediately when the protocol is repeated in parallel via an argument that is similar to the one given in [ORS15]. We start with the description of the building blocks.

Building Blocks.

- A four-round parallel secure OT protocol $(\mathsf{OT}_1, \ldots, \mathsf{OT}_4, \mathsf{out}_{\mathsf{OT}})$ that satisfies the following properties:
  - **IND-Based Receiver Security.** For any malicious adversary $\mathcal{A}$ corrupting the sender and for any two input bits $b_0, b_1 \in \{0,1\}$ of the receiver, we have that the view of the adversary when interacting with a receiver on input b_0 is computationally indistinguishable to its view when interacting with a receiver on input b_1.
  - **1-Rewinding Sender Security.** For every malicious adversary $\mathcal{A}$, corrupting the receiver, there exists an expected polynomial time simulators $\mathsf{Sim}_{\mathcal{S}} = (\mathsf{Sim}^1_{\mathcal{S}}, \mathsf{Sim}^2_{\mathcal{S}})$ such that for every choice of sender's input $x_{\mathcal{S}} = (m_0, m_1)$, we have:

$$\mathsf{Expt}_1(\mathcal{A}, \Pi, x_{\mathcal{S}}) \approx_c \mathsf{Expt}_2(\mathcal{A}, \mathsf{Sim}_{\mathcal{S}}, x_{\mathcal{S}})$$

    where Expt_1 and Expt_2 are defined in Figure 14.

  In Appendix E, we give a proof sketch that the protocol from [ORS15, FMV19] satisfies this property. This protocol makes black-box access to a public key encryption with pseudorandom public keys.

- A 1-rewinding secure extractable commitment scheme $(\mathsf{ECom}_1, \mathsf{ECom}_2, \mathsf{ECom}_3)$.

- A $(\lambda+1)$-out-of-2λ Shamir secret sharing scheme $(\mathsf{Share}, \mathsf{Rec})$ over $\mathbb{F} = \mathsf{GF}[2^\lambda]$. Let $\Psi : \mathbb{F}^{2k} \to \mathbb{F}^{k-1}$ be the linear map such that $\Psi(c) = 0^{\lambda-1}$ iff c is a valid secret sharing of some secret.

- A symmetric key encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.

$\mathsf{Expt}_1(\mathcal{A}, \Pi, x_\mathcal{S}) = 1]$

- Initialize $\mathcal{A}$ with a uniform random tape s.
- $\pi_1 \leftarrow \mathcal{A}(1^\lambda; s)$.
- Choose $r \leftarrow \{0,1\}^\lambda$ uniformly at random and compute $\mathsf{otm}_2 \leftarrow \mathsf{OT}_2(\mathsf{otm}_1, (m_0, m_1); r)$.
- $(\mathsf{otm}_3[0]), (\mathsf{otm}_3[1]) \leftarrow \mathcal{A}(\mathsf{otm}_2; s)$.
- $\mathsf{otm}_4[b] \leftarrow \mathsf{OT}_4(\mathsf{otm}_1, \mathsf{otm}_3[b], (m_0, m_1); r)$ for $b \in \{0,1\}$.
- Output $(s, \mathsf{otm}_1, \mathsf{otm}_2, \{\mathsf{otm}_3[b], \mathsf{otm}_4[b]\}_{b \in \{0,1\}})$.

$\mathsf{Expt}_2(\mathcal{A}, \mathsf{Sim}_\mathcal{S}, x_\mathcal{S})$

- Initialize $\mathcal{A}$ with a uniform random tape s.
- $\mathsf{Sim}^1_\mathcal{S}$ interacts with $\mathcal{A}$ and produces (b, otm_1, sk).
- $\mathsf{Sim}^2_\mathcal{S}$ on input sk and m_b interacts with $\mathcal{A}$ and produces $(\mathsf{otm}_2, \{\mathsf{otm}_3[b], \mathsf{otm}_4[b]\}_{b \in \{0,1\}})$.
- Output $(s, \mathsf{otm}_1, \mathsf{otm}_2, \{\mathsf{otm}_3[b], \mathsf{otm}_4[b]\}_{b \in \{0,1\}})$.

Figure 14: Descriptions of Expt_1 and Expt_2.

Construction.

- **Round-1:** $\mathcal{R}$ on choice bit c does the following:
  1. It chooses a random subset $T_{1-c} \subseteq [2\lambda]$ of size $\lambda/2$ and sets $b_i = c$ for each $i \in [2\lambda] \setminus T_{1-c}$ and sets $b_i = 1 - c$ for each $i \in T_{1-c}$.
  2. For each $i \in [2\lambda]$, it computes $\mathsf{otm}^i_1 \leftarrow \mathsf{OT}_1(1^\lambda, b_i)$.
  3. It sends $\{\mathsf{otm}^i_1\}_{i \in [2\lambda]}$ to $\mathcal{S}$.
- **Round-2:** $\mathcal{S}$ does the following:
  1. For each $b \in \{0,1\}$, it computes $(s^1_b, \ldots, s^{2\lambda}_b) \leftarrow \mathsf{Share}(m_b)$.
  2. For each $i \in [2\lambda]$, $j \in [\lambda]$ and $b \in \{0,1\}$,
     (a) It samples $x^{i,j,1}_b, x^{i,j,2}_b$ uniformly from $\mathbb{F}$.
     (b) For each $k \in [2]$, it computes $u^{j,k}_b := \Psi(x^{1,j,k}_b, \ldots, x^{2\lambda,j,k}_b)$.
     (c) It sets $x^{i,j,3}_b = s^i_b - x^{i,j,1}_b - x^{i,j,2}_b$.
     (d) It samples $r^{i,j,k}_b$ uniformly from $\{0,1\}^*$ for each $k \in [3]$.
     (e) For each $k \in [3]$, it computes $\mathsf{Com}^{i,j,k}_{b,1} := \mathsf{ECom}_1(1^\lambda, x^{i,j,k}_b; r^{i,j,k}_b)$.
  3. For each $i \in [2\lambda]$,
     (a) It chooses two random secret keys sk^i_0, sk^i_1 from $\mathsf{KeyGen}(1^\lambda)$.
     (b) It computes $\mathsf{otm}^i_2 \leftarrow \mathsf{OT}_2(\mathsf{otm}^i_1, sk^i_0, sk^i_1)$.
  4. It sends $\{\mathsf{Com}^{i,j,k}_{b,1}\}_{i \in [2\lambda], j \in [\lambda], k \in [3], b \in \{0,1\}}$, $\{u^{j,k}_b\}_{j \in [\lambda], k \in [2], b \in \{0,1\}}$ and $\{\mathsf{otm}^i_2\}_{i \in [2\lambda]}$ to $\mathcal{R}$.
- **Round-3:** $\mathcal{R}$ does the following:
  1. For each $j \in [\lambda]$, it chooses c_i uniformly from $[3]$.
  2. For each $i \in [2\lambda], j \in [\lambda], k \in [3], b \in \{0,1\}$, it computes $\mathsf{Com}^{i,j,k}_{b,2} \leftarrow \mathsf{ECom}_2(\mathsf{Com}^{i,j,k}_{b,1})$.
  3. For each $i \in [2\lambda]$, it computes $\mathsf{otm}^i_3 \leftarrow \mathsf{OT}_3(\mathsf{otm}^i_2, b_i)$.
  4. It sends $\{\mathsf{otm}^i_3\}_{i \in [2\lambda]}$, $\{\mathsf{Com}^{i,j,k}_{b,2}\}_{i \in [2\lambda], j \in [\lambda], k \in [3], b \in \{0,1\}}$ and $\{c_j\}_{j \in [\lambda]}$.
- **Round-4** : $\mathcal{S}$ does the following:

1. For each $i \in [2\lambda]$,
   (a) It computes $\mathsf{otm}_4^i \leftarrow \mathsf{OT}_4(\mathsf{otm}_1^i, \mathsf{otm}_3^i, sk_0^i, sk_1^i)$.
   (b) It sets $v_b^i := \{r_b^{i,j,k}\}_{j\in[\lambda],k\in[3]}$ for each $b \in \{0,1\}$.
   (c) It computes $ct_b^i = \mathsf{Enc}(sk_b^i, v_b^i)$ for each $b \in \{0,1\}$.
2. For each $i \in [2\lambda], j \in [\lambda], k \in [3], b \in \{0,1\}$, it computes $\mathsf{Com}_{b,3}^{i,j,k} \leftarrow \mathsf{ECom}_2(\mathsf{Com}_{b,2}^{i,j,k}, x_b^{i,j,k}; r_b^{i,j,k})$.
3. It sends $\{\mathsf{otm}_4^i, ct_0^i, ct_1^i\}_{i\in[2\lambda]}$, $\{\mathsf{Com}_{b,3}^{i,j,k}\}_{i\in[2\lambda],j\in[\lambda],k\in[3],b\in\{0,1\}}$, $\{x_b^{i,j,c_j}, r_b^{i,j,c_j}\}_{i\in[2\lambda],j\in[\lambda],b\in\{0,1\}}$.

- **Output:** To compute the output $\mathcal{R}$ does the following:
  1. **Share Validity Check:**
     (a) For each $i \in [2\lambda]$, $j \in [\lambda]$ and $b \in \{0,1\}$, it checks if $\mathsf{Com}_{b,1}^{i,j,c_j}$ and $\mathsf{Com}_{b,3}^{i,j,c_j}$ are consistent with $\{x_b^{i,j,c_j}, r_b^{i,j,c_j}\}$.
     (b) For each $j \in [\lambda]$ and $b \in \{0,1\}$, it sets $u_b^{j,3} = -u_b^{j,1} - u_b^{j,2}$ and checks if $u_b^{j,c_j} = \Psi(x_b^{1,j,c_j}, \ldots, x_b^{2\lambda,j,c_j})$.
  2. **Test Phase:**
     (a) For each $i \in [2\lambda]$,
        i. It recovers $sk_{b_i}^i$ by applying $\mathsf{out}_{\mathsf{OT}}$ on $\mathsf{otm}_2^i, \mathsf{otm}_4^i$, input b_i and the randomness used in generating $\mathsf{otm}_1^i, \mathsf{otm}_3^i$.
        ii. It computes $v_{b_i}^i := \mathsf{Dec}(sk_{b_i}^i, ct_{b_i}^i)$.
        iii. It parses $v_{b_i}^i$ as $\{r_{b_i}^{i,j,k}\}_{j\in[\lambda],k\in[3]}$ and it recovers $\{x_{b_i}^{i,j,k}\}_{j\in[\lambda],k\in[3]}$ from the $\mathsf{Com}_{b_i,1}^{i,j,k}, \mathsf{Com}_{b_i,3}^{i,j,k}$ using $r_{b_i}^{i,j,k}$.
        iv. For each $j \in [\lambda]$, it computes $s_{b_i}^{i,j} = \sum_{k\in[3]} x_{b_i}^{i,j,k}$.
     (b) It chooses a random subset T_c of $[2\lambda] \setminus T_{1-c}$ of size $\lambda/2$.
     (c) It checks if for each $\beta \in \{0,1\}$ and for each $i \in T_\beta$, whether $s_\beta^{i,1} = s_\beta^{i,2} = \cdots = s_\beta^{i,\lambda}$.
     (d) If any of the above checks fail, then it aborts.
  3. **Reconstruction Phase:**
     (a) For each $i \in [2\lambda] \setminus T_{1-c}$, it sets $s_c^i = s_c^{i,1}$ if $s_c^{i,1} = \cdots = s_c^{i,\lambda}$ and otherwise, it sets this value to be $\perp$.
     (b) It runs $\mathsf{Rec}(\{s_b^i\}_{i\in[2\lambda]\setminus T_{1-c}})$ and outputs this value.

D.1 Proof of Security

D.1.1 1-Rewinding Sender Security

We start with the descriptions of the simulators $(\mathsf{Sim}_{\mathcal{S}}^1, \mathsf{Sim}_{\mathcal{S}}^2)$.

Description of $\mathsf{Sim}_{\mathcal{S}}^1$.

1. $\mathsf{Sim}_{\mathcal{S}}^1$ constructs an adversary $\mathcal{A}_1$ that receives the messages for the OT protocol externally and interacts with $\mathcal{A}$ internally in the first three rounds. Specifically, $\mathcal{A}_1$ does the following:
   - It initializes $\mathcal{A}$ and obtains $\{\mathsf{otm}_1^i\}_{i\in[2\lambda]}$ from $\mathcal{A}$. It forwards this message externally.

- On receiving $\{\mathsf{otm}_2^i\}_{i\in[2\lambda]}$ externally, $\mathcal{A}_2$ computes $\mathsf{Com}_{b,1}^{i,j,k} \leftarrow \mathsf{ECom}_1(1^\lambda, \bot)$ for each $i \in [2\lambda], j \in [\lambda], k \in [3]$ and $b \in \{0,1\}$. For each $j \in [\lambda]$ and $k \in [2]$, it computes $u_b^{j,k} := \Psi(x_b^{1,j,k}, \ldots, x_b^{2\lambda,j,k})$ where $x_b^{i,j,k}$ is uniformly chosen for each $i \in [2\lambda]$.
- It runs $\mathcal{A}$ on $\{\mathsf{Com}_{b,1}^{i,j,k}\}_{i\in[2\lambda],j\in[\lambda],k\in[3],b\in\{0,1\}}, \{u_b^{j,k}\}_{j\in[\lambda],k\in[2],b\in\{0,1\}}$ and $\{\mathsf{otm}_2^i\}_{i\in[2\lambda]}$ and receives two sets of third round messages.
- It forwards the OT messages in those two sets externally.

2. $\mathsf{Sim}_{\mathcal{S}}^1$ runs $\mathsf{Sim}_{\mathsf{OT},\mathcal{S}}^1$ on $\mathcal{A}_1$ and obtains $x_{\mathsf{OT}}, sk_{\mathsf{OT}}$ and $\{\mathsf{otm}_1^i\}_{i\in[2\lambda]}$.

3. If x_{OT} corresponds to the special symbol abort, then $\mathsf{Sim}_{\mathcal{S}}^1$ sets $sk = (sk_{\mathsf{OT}},$ random tape of $\mathcal{A}_1)$ and outputs $\mathsf{abort}, sk, \{\mathsf{otm}_1^i\}_{i\in[2\lambda]}$.

4. Otherwise, it parses x_{OT} as $(b_1, \ldots, b_{2\lambda})$. It sets $c = \mathsf{majority}(b_1, \ldots, b_{2\lambda})$. It then runs independent executions of $\mathsf{Sim}_{\mathsf{OT},\mathcal{S}}^1$ on $\mathcal{A}_1$ until it obtains 12λ executions where x_{OT} is not abort. Let T be the number of such executions needed. $\mathsf{Sim}_{\mathcal{S}}^1$ sets $\widetilde{\epsilon} = 12\lambda/T$. It finally outputs $c, sk = ((b_1, \ldots, b_{2\lambda}), sk_{\mathsf{OT}}, \widetilde{\epsilon}, \{\mathsf{otm}_1^i\}_{i\in[2\lambda]})$.

5. $\mathsf{Sim}_{\mathcal{S}}^1$ has an internal step counter and if it runs for more than 2^λ steps, it outputs a special symbol $\mathsf{timeout}$ and aborts.

Description of $\mathsf{Sim}_{\mathcal{S}}^2$.

1. If the output of $\mathsf{Sim}_{\mathcal{S}}^1$ corresponds to the aborting transcript, then $\mathsf{Sim}_{\mathcal{S}}^2$ generates the view of $\mathcal{A}$ in the first three rounds (using the random tape of $\mathcal{A}_1$) and outputs the second and third round messages of the protocol.

2. Otherwise, it obtains m_b from the output of the ideal functionality and sets $m_{1-b} = 0^\lambda$.

3. $\mathsf{Sim}_{\mathcal{S}}^2$ constructs an adversary $\mathcal{A}_2$ that obtains the OT messages externally and interacts with $\mathcal{A}$ internally in the second and third rounds of the protocol as follows:
   - On receiving $\{\mathsf{otm}_2^i\}_{i\in[2\lambda]}$ externally, $\mathcal{A}_2$ samples $\{\mathsf{Com}_{b,1}^{i,j,k}\}_{i\in[2\lambda],j\in[\lambda],k\in[3],b\in\{0,1\}}$ and $\{u_b^{j,k}\}_{j\in[\lambda],k\in[2],b\in\{0,}$ as in the protocol using the newly set values of m_0, m_1.
   - It runs $\mathcal{A}$ on $\{\mathsf{Com}_{b,1}^{i,j,k}\}_{i\in[2\lambda],j\in[\lambda],k\in[3],b\in\{0,1\}}, \{u_b^{j,k}\}_{j\in[\lambda],k\in[2],b\in\{0,1\}}$ and $\{\mathsf{otm}_2^i\}_{i\in[2\lambda]}$ and receives two sets of third round messages.
   - It forwards the OT messages in those two sets of third round messages externally.

4. For each $i \in [2\lambda]$, $\mathsf{Sim}_{\mathcal{S}}^2$ samples two secret keys sk_0^i, sk_1^i from $\mathsf{KeyGen}(1^\lambda)$.

5. $\mathsf{Sim}_{\mathcal{S}}^2$ runs $\mathsf{Sim}_{\mathsf{OT},\mathcal{S}}^2$ on $\mathcal{A}_2$ with sk_{OT} and $\{sk_{b_i}^i\}_{i\in[2\lambda]}$ as inputs for $\lambda^2/\widetilde{\epsilon}$ independent executions. If in each of these executions, $\mathsf{Sim}_{\mathsf{OT},\mathcal{S}}^2$ outputs the special symbol fail, then $\mathsf{Sim}_{\mathcal{S}}^2$ outputs fail and aborts.

6. Otherwise, $\mathsf{Sim}_{\mathcal{S}}^2$ obtains the second round OT protocol message, the two sets of third and fourth round OT protocol messages from $\mathsf{Sim}_{\mathsf{OT},\mathcal{S}}^2$.

7. From the random tape of $\mathcal{A}_2$, $\mathsf{Sim}_{\mathcal{S}}^2$ recovers the rest of the second and third round messages of the protocol in its interaction with $\mathcal{A}$ along with $\{r_{b_i}^{i,j,k}\}_{i\in[2\lambda],j\in[\lambda],k\in[3]}$ which denotes the randomness used in generating $\{\mathsf{Com}_{b_i}^{i,j,k}\}_{i\in[2\lambda],j\in[\lambda],k\in[3]}$.

8. $\mathsf{Sim}^2_{\mathcal{S}}$ computes $ct^i_\beta = \mathsf{Enc}(sk^i_\beta, \{r^{i,j,k}_{b_i}\}_{j\in[\lambda],k\in[3]})$ for each $i \in [2\lambda]$ and $\beta \in \{0,1\}$.
9. It generates the rest of the last round protocol messages using the random tape of $\mathcal{A}_2$ and outputs the second, two sets of third and fourth round messages of the protocol.

Running time of $\mathsf{Sim}^1_{\mathcal{S}}$ and $\mathsf{Sim}^2_{\mathcal{S}}$. Let ϵ be the probability that $\mathsf{Sim}^1_{\mathsf{OT},\mathcal{S}}$ outputs $x_{\mathsf{OT}} \neq \mathsf{abort}$ in the description of $\mathsf{Sim}^1_{\mathcal{S}}$. Then the expected running time of $\mathsf{Sim}^1_{\mathcal{S}}$ is given by $\mathsf{poly}(\lambda)+\mathsf{poly}(\lambda)\epsilon(12\lambda/\epsilon) = \mathsf{poly}(\lambda)$. Via a standard argument (see [GK96a]), we can show that $\widetilde{\epsilon}$ is within a factor of 2 of the value of ϵ except with probability $2^{-\lambda}$. Thus, the expected running time of $\mathsf{Sim}^2_{\mathcal{S}}$ is given by $\mathsf{poly}(\lambda) + \mathsf{poly}(\lambda)\epsilon(O(\lambda^2/\epsilon) + 2^\lambda(1/2^\lambda)) = \mathsf{poly}(\lambda)$.

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

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the output of the real experiment.
- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we construct an adversary $\mathcal{A}'$ that receives the OT protocol messages externally and interacts with $\mathcal{A}$ in the first three rounds of the protocol as follows:
  - It initializes $\mathcal{A}$ and obtains $\{\mathsf{otm}^i_1\}_{i\in[2\lambda]}$ from $\mathcal{A}$. It forwards this message externally.
  - On receiving $\{\mathsf{otm}^i_2\}_{i\in[2\lambda]}$ externally, $\mathcal{A}_2$ samples $\{\mathsf{Com}^{i,j,k}_{b,1}\}_{i\in[2\lambda],j\in[\lambda],k\in[3],b\in\{0,1\}}$ and $\{u^{j,k}_b\}_{j\in[\lambda],k\in[2],b\in\{0,$ as in the protocol using the honest parties inputs m_0, m_1.
  - It runs $\mathcal{A}$ on $\{\mathsf{Com}^{i,j,k}_{b,1}\}_{i\in[2\lambda],j\in[\lambda],k\in[3],b\in\{0,1\}}$, $\{u^{j,k}_b\}_{j\in[\lambda],k\in[2],b\in\{0,1\}}$ and $\{\mathsf{otm}^i_2\}_{i\in[2\lambda]}$ and receives two sets of third round messages.
  - It forwards the OT messages in those two sets externally.

  We generate the OT protocol messages in this hybrid by running $\mathsf{Sim}^1_{\mathsf{OT},\mathcal{S}}, \mathsf{Sim}^2_{\mathsf{OT},\mathcal{S}}$ on $\mathcal{A}_2$. In Claim D.2, we show that $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$ from the 1-rewinding sender security property of the OT protocol.
- $\underline{\mathsf{Hyb}_2}$: In this hybrid,
  - We estimate $\widetilde{\epsilon}$ as in the simulation using $\mathcal{A}'$. We output the special symbol $\mathsf{timeout}$, if we run for 2^λ steps.
  - We run $\mathsf{Sim}^2_{\mathsf{OT},\mathcal{S}}$ on $\mathcal{A}'$ for $\lambda^2/\widetilde{\epsilon}$ independent executions and we output the special symbol fail, if $\mathsf{Sim}^2_{\mathsf{OT},\mathcal{S}}$ outputs fail, in each of these executions. We output the special symbol $\mathsf{timeout}$, if we run for 2^λ steps.

  Via an identical argument to Claim C.4, we can show that $\mathsf{Hyb}_1 \approx_s \mathsf{Hyb}_2$.
- $\underline{\mathsf{Hyb}_3 :}$ In this hybrid, we run $\mathsf{Sim}^1_{\mathsf{OT},\mathcal{S}}$ on the adversary $\mathcal{A}_1$ (defined in the simulation), instead of running it on $\mathcal{A}'$ as in the previous hybrid.

  Via an identical argument in Claim C.5, we can show that $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$ from the 1-rewinding security of the extractable commitment scheme.
- $\underline{\mathsf{Hyb}_4}$: In this hybrid, we generate $ct^i_{1-b_i}$ for each $i \in [2\lambda]$ as $\mathsf{Enc}(sk^i_{1-b_i}, \{r^{i,j,k}_{b_i}\}_{j\in[\lambda],k\in[3]})$.

  In Claim D.3, we show that $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$ from the semantic security of the encryption scheme.

- $\underline{\mathsf{Hyb}_5}$: In this hybrid, we run $\mathsf{Sim}^2_{\mathsf{OT},\mathcal{S}}$ on the adversary $\mathcal{A}_2$ instead of running it on $\mathcal{A}'$ as in the previous hybrid.

  In Claim D.4, we show that $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$ from the 1-rewinding security of the extractable commitment scheme and the perfect secrecy of Shamir sharing.

Claim D.2. *Assuming the 1-rewinding sender security property of the OT protocol, we have* $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$.

Proof. Assume for the sake of contradiction that there exists a distinguisher D that can distinguish between Hyb_0 and Hyb_1 with non-negligible advantage. We will use D to construct an adversary $\mathcal{B}$ that breaks the 1-rewinding sender security of the OT protocol.

- For each $i \in [2\lambda]$, $\mathcal{B}$ samples sk_0^i, sk_1^i from $\mathsf{KeyGen}(1^\lambda)$ and sends $\{sk_0^i, sk_1^i\}_{i\in[m]}$ as the challenge sender inputs to the external challenger.
- $\mathcal{B}$ constructs $\mathcal{A}'$ as in the description of Hyb_1 and forwards the messages from the external challenger to $\mathcal{A}'$ and vice-versa.
- The external challenger finally outputs the view of $\mathcal{A}'$. $\mathcal{B}$ uses this to generate the final round message of the protocol and generate the view of $\mathcal{A}$.
- $\mathcal{B}$ finally runs D on the above generated view of $\mathcal{A}$ and outputs whatever D outputs.

Note that if the view of $\mathcal{A}'$ is generated using the real algorithms then the inputs to D are identically distributed to Hyb_0. Otherwise, they are distributed identically to Hyb_1. Thus, if D can distinguish between Hyb_0 and Hyb_1 with non-negligible advantage, then $\mathcal{A}$ breaks the 1-rewinding sender security property of the OT protocol and this is a contradiction. □

Claim D.3. *Assuming the semantic security of the encryption scheme, we have* $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$.

Proof. 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 will use D to construct an adversary $\mathcal{B}$ that breaks the semantic security of the encryption scheme.

- For each $i \in [2\lambda]$, $\mathcal{A}$ provides $\{r_{b_i}^{i,j,k}\}_{j\in[\lambda],k\in[3]}$ as the left challenge message and $\{r_{1-b_i}^{i,j,k}\}_{j\in[\lambda],k\in[3]}$ as the right challenge message. It receives $ct_{1-b_i}^i$ from the challenger.
- It generates the rest of the protocol messages as in Hyb_3. Note that this does not require knowledge of $sk_{1-b_i}^i$.
- It generates the view of the adversary $\mathcal{A}$ and finally runs D on this view and outputs whatever D outputs.

Note that if the challenge ciphertexts correspond to the encryption of the left message, then the inputs to D are identical to Hyb_4. Otherwise, the inputs are identical to Hyb_3. Thus, if D can distinguish between Hyb_3 and Hyb_4 with non-negligible advantage, then $\mathcal{B}$ can break the semantic security of the encryption scheme and this is a contradiction. □

Claim D.4. *Assuming the 1-rewinding sender security of the extractable commitment scheme, we have* $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$.

Proof. Let $c = \mathsf{majority}(b_1, \ldots, b_{2\lambda})$ and let T_{1-c} be the set of indices such that for each $i \in T_{1-c}$, $b_i = 1 - c$. By definition, $|T_{1-c}| \leq \lambda$. For each $j \in [\lambda]$, we define an adversary $\mathcal{A}'_j$ that does the following:

- On receiving $\{\mathsf{otm}_2^i\}_{i\in[2\lambda]}$ externally, $\mathcal{A}'_j$ samples s^i_{1-b} for each $i \in T_{1-c}$ uniformly at random from $\mathbb{F}$. It sets $\overline{s}^i_{1-b} = s^i_{1-b}$ and generates $(s^1_{1-b}, \ldots, s^{2\lambda}_{1-b}) \leftarrow \mathsf{Share}(m_{1-b})|\{s^i_b\}_{i\in T_{1-c}}$ and $(\overline{s}^1_{1-b}, \ldots, \overline{s}^{2\lambda}_{1-b}) \leftarrow \mathsf{Share}(0^\lambda)|\{\overline{s}^i_b\}_{i\in T_{1-c}}$.
- It then samples $\{\mathsf{Com}^{i,j',k}_{1-b,1}\}_{i\in[2\lambda],j'\leq j,k\in[3]}$ and $\{u^{j',k}_{1-b}\}_{j'\leq j,k\in[2]}$ using $(\overline{s}^1_{1-b}, \ldots, \overline{s}^{2\lambda}_{1-b})$ and samples $\{\mathsf{Com}^{i,j',k}_{1-b,1}\}_{i\in[2\lambda],j'>j,k\in[3]}$ and $\{u^{j',k}_{1-b}\}_{j'>j,k\in[2]}$ using $(s^1_{1-b}, \ldots, s^{2\lambda}_{1-b})$.
- It runs $\mathcal{A}$ on $\{\mathsf{Com}^{i,j,k}_{b,1}\}_{i\in[2\lambda],j\in[\lambda],k\in[3],b\in\{0,1\}}$, $\{u^{j,k}_b\}_{j\in[\lambda],k\in[2],b\in\{0,1\}}$ and $\{\mathsf{otm}_2^i\}_{i\in[2\lambda]}$ and receives two sets of third round messages.
- It forwards the OT messages in those two sets externally.

We consider a sequence of hybrids $\mathsf{Hyb}_4 \equiv \mathsf{Hyb}_{4,0}, \ldots, \mathsf{Hyb}_{4,\lambda} \equiv \mathsf{Hyb}_5$ where the only difference between $\mathsf{Hyb}_{4,j-1}$ and $\mathsf{Hyb}_{4,j}$ (for each $j \in [\lambda]$) is that in $\mathsf{Hyb}_{4,j}$, we run $\mathsf{Sim}^2_{\mathsf{OT},\mathcal{S}}$ on $\mathcal{A}'_j$ whereas in $\mathsf{Hyb}_{4,j-1}$, we run it on $\mathcal{A}'_{j-1}$.

We now show that for each $j \in [\lambda]$, $\mathsf{Hyb}_{4,j-1} \approx_c \mathsf{Hyb}_{4,j}$ from the 1-rewinding security of the extractable commitment scheme.

We consider a couple of intermediate hybrids.

- <u>$\mathsf{Hyb}'_{4,j-1}$</u> : In this hybrid, we define an adversary $\mathcal{C}'_{j-1}$ that does the following:
  - On receiving $\{\mathsf{otm}_2^i\}_{i\in[2\lambda]}$ externally, $\mathcal{C}'_{j-1}$
    1. Sets $\mathsf{count} = 0$.
    2. While($\mathsf{count} \leq \lambda$)
       (a) It chooses a random $g_j \leftarrow [3]$.
       (b) It samples s^i_{1-b} for each $i \in T_{1-c}$ uniformly at random from $\mathbb{F}$. It sets $\overline{s}^i_{1-b} = s^i_{1-b}$ and generates $(s^1_{1-b}, \ldots, s^{2\lambda}_{1-b}) \leftarrow \mathsf{Share}(m_{1-b})|\{s^i_b\}_{i\in T_{1-c}}$ and $(\overline{s}^1_{1-b}, \ldots, \overline{s}^{2\lambda}_{1-b}) \leftarrow \mathsf{Share}(0^\lambda)|\{\overline{s}^i_b\}_{i\in T_{1-c}}$.
       (c) It then samples $\{\mathsf{Com}^{i,j',k}_{1-b,1}\}_{i\in[2\lambda],j'\leq j,k\in[3]}$ and $\{u^{j',k}_{1-b}\}_{j'\leq j-1,k\in[2]}$ using $(\overline{s}^1_{1-b}, \ldots, \overline{s}^{2\lambda}_{1-b})$ and samples $\{\mathsf{Com}^{i,j',k}_{1-b,1}\}_{i\in[2\lambda],j'\geq j,k\in[3]}$ and $\{u^{j',k}_{1-b}\}_{j'\geq j,k\in[2]}$ using $(s^1_{1-b}, \ldots, s^{2\lambda}_{1-b})$.
       (d) It runs $\mathcal{A}$ on $\{\mathsf{Com}^{i,j,k}_{b,1}\}_{i\in[2\lambda],j\in[\lambda],k\in[3],b\in\{0,1\}}$, $\{u^{j,k}_b\}_{j\in[\lambda],k\in[2],b\in\{0,1\}}$ and $\{\mathsf{otm}_2^i\}_{i\in[2\lambda]}$ and receives two sets of third round messages.
       (e) If the c_j received $\mathcal{A}$ in any of the two sets of third round messages is equal to g_j, then it increments count and goes to the beginning of the while loop. Otherwise, it forwards the OT messages in those two sets externally.
    3. If $\mathsf{count} = \lambda + 1$, then it aborts.

  We now argue that $\mathsf{Hyb}'_{4,j-1}$ is statistically close to $\mathsf{Hyb}_{4,j-1}$. Note that g_j is distributed uniformly given the view of the adversary $\mathcal{A}$. Thus, in each execution of the while loop, the probability that g_j is equal to one of the c_js in the two sets of third round messages is at most $2/3$. Thus, in each of λ independent executions of the while loop g_j is equal to one of the c_js is at most $(2/3)^\lambda = \mathsf{negl}(\lambda)$.

- $\mathsf{Hyb}'_{4,j}$: In this hybrid, we construct an adversary $\mathcal{C}'_j$ that does is similar to $\mathcal{C}'_{j-1}$, except that in each execution of the while loop, it samples $\{\mathsf{Com}^{i,j',k}_{1-b,1}\}_{i\in[2\lambda],j'\leq j,k\in[3]}$ and $\{u^{j',k}_{1-b}\}_{j'\leq j,k\in[2]}$ using $(\overline{s}^1_{1-b},\ldots,\overline{s}^{2\lambda}_{1-b})$ and samples $\{\mathsf{Com}^{i,j',k}_{1-b,1}\}_{i\in[2\lambda],j'>j,k\in[3]}$ and $\{u^{j',k}_{1-b}\}_{j'>j,k\in[2]}$ using $(s^1_{1-b},\ldots,s^{2\lambda}_{1-b})$.

  We now argue that $\mathsf{Hyb}'_{4,j-1} \approx_c \mathsf{Hyb}'_{4,j}$ from the 1-rewinding sender security of the extractable commitment scheme.

  Assume for the sake of contradiction that there exists a distinguisher D that can distinguish between $\mathsf{Hyb}'_{4,j}$ and $\mathsf{Hyb}'_{4,j-1}$ with non-negligible advantage. We now construct an adversary $\mathcal{B}$ that breaks the 1-rewinding sender security of the extractable commitment scheme.

  $\mathcal{B}$ constructs an adversary $\mathcal{C}$ that is similar to $\mathcal{C}'_{j-1}$ except that:

  1. In each execution of the while loop, it samples a random element $g_j \leftarrow [3]$.
  2. For each $k \notin [3]\setminus\{g_j\}$, it samples $(x^{1,j,k}_{1-b},\ldots,x^{2\lambda,j,k}_{1-b})$ uniformly at random.
  3. It then computes $x^{i,j,g_j}_{1-b} = s^i_{1-b} - \sum_{k\in[3]\setminus\{g_j\}} x^{i,j,k}_{1-b}$ and $\overline{x}^{i,j,g_j}_{1-b} = \overline{s}^i_{1-b} - \sum_{k\in[3]\setminus\{g_j\}} x^{i,j,k}_{1-b}$ for each $i\in[2\lambda]$. If $g_j \in [2]$, then it sets $u^{j,g_j}_{1-b} = -\sum_{k\in[3]\setminus\{g_j\}} \Psi(x^{1,j,k}_{1-b},\ldots,x^{2\lambda,j,k}_{1-b})$.
  4. It interacts with the external challenger and provides $\{x^{i,j,g_j}_{1-b}\}_{i\in[2\lambda]}$ as the left message and $\{\overline{x}^{i,j,g_j}_{1-b}\}_{i\in[2\lambda]}$ as the right message.
  5. It receives $\{\mathsf{Com}^{i,j,g_j}_{b,1}\}_{i\in[2\lambda]}$ from the external challenger and generates the rest of the second round messages exactly as $\mathcal{C}'_{j-1}$.
  6. The rest of the steps are same as in $\mathcal{C}'_{j-1}$.

  $\mathcal{B}$ runs $\mathsf{Sim}^2_{\mathsf{OT},\mathcal{S}}$ on $\mathcal{C}$ as in $\mathsf{Hyb}'_{4,j-1}$ and records the interaction $\mathcal{C}$ has with $\mathcal{A}$. When $\mathsf{Sim}^2_{\mathsf{OT},\mathcal{S}}$ generates the second, two sets of third and fourth round OT messages, $\mathcal{B}$ retrieves the two sets of second round extractable commitment messages $\{\mathsf{Com}^{i,j,g_j}_{1-b,2}[\beta]\}_{i\in[2\lambda],\beta\in\{0,1\}}$ from $\mathcal{C}'$s interaction with $\mathcal{A}$ in the third round of the protocol. It sends this to the external challenger and obtains the two sets of corresponding third round messages. It generates the rest of the final round messages as in $\mathsf{Hyb}'_{4,j-1}$. $\mathcal{B}$ finally runs D on the view of the adversary $\mathcal{A}$ and then outputs whatever D outputs.

  Note that if commitments generated by the external challenger correspond to the left message, then the inputs to D are identical to output of $\mathsf{Hyb}'_{4,j-1}$. Otherwise, they are identical to the output of $\mathsf{Hyb}'_{4,j}$. Thus, if D can distinguish between $\mathsf{Hyb}'_{4,j-1}$ and $\mathsf{Hyb}'_{4,j}$ with non-negligible advantage then $\mathcal{B}$ can break the 1-rewinding sender security of the extractable commitment scheme and this is a contradiction.

We can show via an identical argument to the proof that $\mathsf{Hyb}'_{4,j-1}$ is statistically close to $\mathsf{Hyb}_{4,j-1}$ that $\mathsf{Hyb}'_{4,j}$ is statistically close to $\mathsf{Hyb}_{4,j}$. This completes the proof of the claim. □

D.1.2 Receiver Security.

We start with the descriptions of $(\mathsf{Sim}^1_{\mathcal{R}},\mathsf{Sim}^2_{\mathcal{R}})$.

Description of $\mathsf{Sim}^1_{\mathcal{R}}$.

1. $\mathsf{Sim}^1_{\mathcal{R}}$ follows the honest receiver protocol using the input $c = 0$ and plays the first four rounds of the protocol.

2. $\mathsf{Sim}^1_{\mathcal{R}}$ rewinds the third and the fourth rounds and gives different second round extractable commitment challenges and extracts $\{x_0^{i,j,k}, x_1^{i,j,k}\}_{i\in[2\lambda],j\in[\lambda],k\in[3]}$.

3. For each $i \in [2\lambda]$ and $j \in [\lambda]$, it computes $s_b^{i,j} := \sum_{k\in[3]} x_b^{i,j,k}$.

4. For each $i \in [2\lambda] \setminus T_1$ and $b \in \{0,1\}$, it sets $s_b^i = s_b^{i,1}$ if $s_b^{i,1} = s_b^{i,2} \cdots = s_b^{i,\lambda}$ and otherwise, it sets this value to be $\perp$.

5. It sets $m_b = \mathsf{Rec}(\{s_b^i\}_{i\in[2\lambda]\setminus T_1})$ for each $b \in \{0,1\}$.

6. It outputs (m_0, m_1), the messages in the first two rounds and sets sk to be its random tape.

Description of $\mathsf{Sim}^2_{\mathcal{R}}$.

- $\mathsf{Sim}^2_{\mathcal{R}}$ plays the third and fourth round of the protocol using the random tape given as part of sk.
- It performs all the checks that an honest receiver does in the Share validity check and the Test phase. If any of the checks fail, it instructs the ideal functionality to abort the protocol.
- If all the checks pass, then it instructs the ideal functionality to deliver the outputs to the honest receiver.

Proof of Indistinguishability. We show that the view of the adversary and the output of the honest receiver in the real and the ideal experiments are computationally indistinguishable via a hybrid argument.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the output of the real experiment.
- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we rewind the third and the fourth rounds of the protocol and extract $\{x_0^{i,j,k}, x_1^{i,j,k}\}_{i\in[2\lambda],j\in[\lambda],k\in[3]}$ from the extractable commitment scheme. We perform the share validity check and if it aborts then we instruct the honest receiver to abort. In the test phase, we choose a random subset T_c of $[2\lambda] \setminus T_{1-c}$ of size $\lambda/2$ and recover $\{sk_{b_i}^i\}_{i\in T_0\cup T_1}$ from $\mathsf{out}_{\mathsf{OT}}$. We then recover $\{v_{b_i}^i\}_{i\in T_0\cup T_1}$ by decrypting $\{ct_{b_i}^i\}_{i\in T_0\cup T_1}$ and perform the rest of the checks in the test phase. In the reconstruction phase, we use the extracted values instead of the values recovered from the OT computation to reconstruct m_c.

  In Claim D.5, we show that $\mathsf{Hyb}_0 \approx_s \mathsf{Hyb}_1$. Notice that if the honest receiver's choice bit $c = 0$, then Hyb_1 is identical to the output of the ideal experiment.
- $\underline{\mathsf{Hyb}_2}$: In this hybrid, for each $i \in [2\lambda] \setminus \{T_0 \cup T_1\}$, we change b_i from c to $1 - c$. Note that this hybrid is identical to the output of the ideal experiment if the choice bit c of the honest receiver is 1.

  In Claim D.6, we show that $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$ from the Ind-based receiver security of the OT protocol.

Claim D.5. $\mathsf{Hyb}_0 \approx_s \mathsf{Hyb}_1$.

Proof. We first argue that with overwhelming probability, there exists at least one $j \in [\lambda]$ such that $\{x_b^{1,j,k}, \ldots, x_b^{2\lambda,j,k}\}_{k \in [3]}$ form a 3-out-of-3 secret sharing of the output of Share on some secret. Suppose this is not the case, then for each $j \in [\lambda]$ there exists at least one $k_j \in [3]$ such that if $c_j = k_j$, then the share validity check outputs abort. Now, since c_j is chosen uniformly at random for each $j \in [\lambda]$, the probability that $c_j \neq k_j$ for each $j \in [\lambda]$ is at most $(2/3)^\lambda$.

Next, we argue that if there exists a subset $C \subseteq [2\lambda] \setminus T_{1-c}$ of size $\lambda/8$ such that for each $i \in C$ either:

1. There exists some $j \in [\lambda], k \in [3]$ such that $\{\mathsf{Com}_{b,1}^{i,j,k}, \mathsf{Com}_{b,3}^{i,j,k}\}$ is an improperly computed extractable commitment.

2. There exists a $j \in [\lambda]$ such that $s_b^{i,j}$ recovered in the test phase is not equal to $s_b^{i,1}$.

Then, the test phase aborts with overwhelming probability. This is because if $T_c \cap C \neq \emptyset$, then the test phase aborts and via an identical argument to the proof of Claim 6.7, we can show that this happens with probability at least $1 - 2^{-O(\lambda)}$. Thus, it now follows from the error correcting properties of Shamir secret sharing (which can recover upto $\lambda/4$ errors) that the output of the reconstruction phase in Hyb_1 is identical to Hyb_2 conditioned on the share validity check and the test phase not aborting. □

Claim D.6. *Assuming the Ind-based receiver security of the OT protocol, we have* $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$.

Proof. Assume for the sake of contradiction that there exists a distinguisher D that can distinguish between Hyb_1 and Hyb_2 with non-negligible advantage. We will use D to construct an adversary $\mathcal{B}$ that breaks the Ind-based receiver security of the OT protocol.

- $\mathcal{B}$ chooses random subset $T_{1-c} \subseteq [2\lambda]$ of size $\lambda/2$ and then chooses a random subset $T_c \subseteq [2\lambda] \setminus T_{1-c}$ of size $\lambda/2$.
- For each $i \in T_c$, it uses $b_i = c$ and for each $i \in T_{1-c}$, it uses $b_i = 1 - c$.
- For each $i \in [2\lambda] \setminus \{T_0 \cup T_1\}$, $\mathcal{B}$ interacts with the external challenger and provides $(c, 1-c)$ as the two challenge choice bits. It receives $\{\mathsf{otm}_1^i\}_{i \in [2\lambda] \setminus \{T_0 \cup T_1\}}$.
- It generates the rest of the first round message as in Hyb_1.
- On receiving the second round message from the adversary, $\mathcal{B}$ forwards $\{\mathsf{otm}_2^i\}_{i \in [2\lambda] \setminus \{T_0 \cup T_1\}}$ to the external challenger and obtains $\{\mathsf{otm}_3^i\}_{i \in [2\lambda] \setminus \{T_0 \cup T_1\}}$.
- It generates the rest of the third round messages as in Hyb_2 and computes the output m_c as in Hyb_2.
- $\mathcal{B}$ runs D on the view of the adversary and the output of the honest receiver and outputs whatever D outputs.

Note that if the OT messages generated by the challenger contains the choice bit c, then the inputs to D are identically distributed to Hyb_1. Otherwise, they are distributed identically to Hyb_2. Thus, if D can distinguish between the outputs of Hyb_1 and Hyb_2 with non-negligible advantage, then $\mathcal{B}$ breaks the Ind-based receiver security of the OT protocol which is a contradiction. □

Parallel Security. The parallel security follows immediately via a similar argument as in [ORS15].

E Proof Sketch for 1-Rewinding Sender Security of [FMV19]

In this subsection, we give a sketch of the proof that the protocol from [FMV19] satisfies 1-rewinding sender security. We begin with the description of the building blocks and the protocol. The description of the building blocks and the protocol are taken verbatim from [FMV19].

Building Blocks.

- A commit-and-prove protocol $\Pi_{co} := (P_0, P_1, V_0, V_1)$ from [ORS15] with the following syntax. (i) The randomized algorithm P_0 takes a bit d and a string m_d and returns a string $\gamma \in \{0,1\}^*$ and auxiliary state information $\alpha \in \{0,1\}^*$. The randomized algorithm V_0 returns a random string $\beta \leftarrow \mathcal{B}$. The randomized algorithm P_1 takes $(\alpha, \beta, \gamma, m_{1-d})$ and returns a string $\delta \in \{0,1\}^*$; (iv) The deterministic algorithm V_1 takes a transcript $(\gamma, \beta, (\delta, m_0, m_1))$ and outputs a bit. We need this protocol to satisfy the following properties:
  - **Binding Property.** For every PPT malicious prover $P^* = (P_0^*, P_1^*)$, there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ such that probability that the following experiment outputs 1 is $\nu(\lambda)$:[15]
    1. $(\gamma, \alpha) \leftarrow P_0^*(1^\lambda)$.
    2. $\beta \leftarrow V_0(1^\lambda)$.
    3. $(\delta, m_0, m_1), (\delta', m_0', m_1') \leftarrow P_1^*(\alpha, \beta)$
    4. Output $V_1(\gamma, \beta, (\delta, m_0, m_1)) \wedge V_1(\gamma, \beta, (\delta', m_0', m_1')) \wedge \wedge\, m_0' \neq m_0 \wedge m_1' \neq m_1$.
  - **Existence of Committing Branch**: For every PPT malicious prover $P^* = (P_0^*, P_1^*)$, there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ such that the probability that the following experiment outputs 1 is at most $\nu(\lambda)$.
    1. $(\gamma, \alpha) \leftarrow P_0^*(1^\lambda)$.
    2. $\beta, \beta' \leftarrow V_0(1^\lambda)$.
    3. $(\delta, m_0, m_1) \leftarrow P_1^*(\alpha, \beta)$
    4. $(\delta', m_0', m_1') \leftarrow P_1^*(\alpha, \beta')$.
    5. Output $V_1(\gamma, \beta, (\delta, m_0, m_1)) \wedge V_1(\gamma, \beta', (\delta', m_0', m_1')) \wedge m_0' \neq m_0 \wedge m_1' \neq m_1$.
  - **Committing Branch Indistinguishability:** For all PPT malicious verifiers V^* and for all messages $(m_0, m_1) \in \{0,1\}^*$, we have that the view of V^* when interacting with an honest prover on input $(m_0, m_1, 0)$ is computationally indistinguishable to its view when interacting with an honest prover on input $(m_0, m_1, 1)$.
- A public-key encryption $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with pseudorandom public keys and the public keys are strings in $\{0,1\}^\lambda$.

Construction. We now describe the protocol from [FMV19].

- **Round-1:** $\mathcal{R}$ does the following:
  1. It chooses a random string $s_{1-b} \leftarrow \{0,1\}^\lambda$.
  2. It runs $P_0(1^\lambda, 1-b, s_{1-b})$ to obtain (α, γ).

[15]Though not explicitly mentioned, we note that the [ORS15] satisfies this binding property.

3. It sends γ to $\mathcal{S}$.

- **Round-2:** $\mathcal{S}$ does the following:
  1. It samples two random strings $r_0, r_1 \leftarrow \{0,1\}^\lambda$.
  2. It samples $\beta \leftarrow V_0(1^\lambda)$.
  3. It sends (r_0, r_1, β).
- **Round-3:** $\mathcal{R}$ does the following:
  1. It samples $(pk_b, sk_b) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
  2. It sets $s_b = pk_b - r_b$.
  3. It computes $(\delta, s_0, s_1) \leftarrow P_1(\alpha, \beta, \gamma, s_b)$.
  4. It sends (δ, s_0, s_1).

  Round-4: $\mathcal{S}$ does the following:
  1. It checks if $V_1(\beta, \gamma, (\delta, s_0, s_1)) = 1$ and otherwise, it aborts.
  2. Else, it computes $pk_0 = r_0 \oplus s_0$ and $pk_1 = r_1 \oplus s_1$.
  3. It then computes $ct_0 \leftarrow \mathsf{Enc}(pk_0, m_0)$ and $ct_1 \leftarrow \mathsf{Enc}(pk_1, m_1)$.
  4. It then sends ct_0, ct_1 to $\mathcal{R}$.
- **Output:** To compute the output, $\mathcal{R}$ decrypts ct_b using sk_b and recovers m_b.

Proof Sketch for 1-Rewinding Sender Security. It follows from the binding property that for any two different third round messages of $\mathcal{R}$, the message s_{1-b} that corresponds to the committing branch must be the same. Therefore, even if $\mathcal{R}$ rewinds the third round once, it follows from the semantic security of the public key encryption under pk_{1-b} that the message m_{1-b} must be hidden. This is formalized using a simulator that generates ct_{1-b} in the main thread as well as the rewind thread as an encryption of $\perp$.

F Five-Round MPC protocol over Point-to-Point Channels

In this section, we give a modification of the protocol from Section 6 that works over point-to-point channels instead of the broadcast channels. One of the key ingredients in this modification is a special Conditional Disclosure of Secrets (CDS) protocol.

F.1 Special CDS Protocol

In this special CDS protocol, there are two senders S_1 and S_2 and one receiver R. The input of the sender S_i is given by $(x_i, y_i) \in \{0,1\}^*$ for each $i \in [2]$. The senders S_1 and S_2 share some common randomness r that is unknown to the receiver. We want S_1 and S_2 to send a single message to the receiver R such that if $x_1 = x_2$ then it can obtain (y_1, y_2). Otherwise, the receiver learns no information about (y_1, y_2). Formally, this special CDS protocol has the following syntax:

- $\mathsf{CDS}_1(i, (x_i, y_i), r)$: It is a PPT algorithm that takes the index $i \in [2]$ of the party, its input (x_i, y_i) and the shared random string r and outputs cds_1^i.

- $\mathsf{CDS}_2(\mathsf{cds}_1^1, \mathsf{cds}_1^2, (x_1, x_2))$: It is a deterministic algorithm that takes $\mathsf{cds}_1^1, \mathsf{cds}_1^2, (x_1, x_2)$ as input and outputs either (y_1, y_2) or $\perp$.

We require the CDS protocol to satisfy the following properties.

- **Correctness:** For every input (x_1, y_1), (x_2, y_2) such that $x_1 = x_2$ and the shared random string r, we have:
$$\Pr[\mathsf{CDS}_2(\mathsf{cds}_1^1, \mathsf{cds}_1^2, (x_1, x_2)) = (y_1, y_2)] = 1$$
where $\mathsf{cds}_1^i \leftarrow \mathsf{CDS}_1(i, (x_i, y_i), r)$ for each $i \in [2]$.

- **Security**: There exists a simulator Sim such that for any inputs (x_1, y_1), (x_2, y_2) such that $x_1 \neq x_2$, we have:
$$\{\mathsf{cds}_1^1, \mathsf{cds}_1^2\} \approx_c \{\mathsf{Sim}(x_1, x_2)\}$$
where $\mathsf{cds}_1^i \leftarrow \mathsf{CDS}_1(i, (x_i, y_i), r)$ for each $i \in [2]$.

We now give the description of such a protocol. Let us assume that each x_i, y_i belongs to some finite Field $\mathbb{F}$ of size greater than 2^λ. Let the shared random string r be parsed as (a, b, c) where $a, b, c \in \mathbb{F}$. Let us assume that there exists a symmetric key encryption scheme $\mathsf{Enc}, \mathsf{Dec}$ where the key space and the message space are $\mathbb{F}$. Further, the keys are uniformly chosen random elements in $\mathbb{F}$.

- $\mathsf{CDS}_1(i, (x_i, y_i), r)$: If $i = 1$, compute $k_1 = a \cdot x_1 + b + c$. Else, compute $k_2 = a \cdot x_2 + b$. Compute $\mathsf{ct}_i = \mathsf{Enc}_c(y_i)$. Output (k_i, ct_i).

- $\mathsf{CDS}_2(\mathsf{cds}_1^1, \mathsf{cds}_1^2, (x_1, x_2))$: Compute $y_i := \mathsf{Dec}_{k_1 - k_2}(\mathsf{ct}_i)$ for each $i \in [2]$ and output (y_1, y_2).

The correctness is easy to observe and to show security notice that if $x_1 \neq x_2$, then conditioned on k_2, $ax_1 + b$ is uniformly distributed in the field $\mathbb{F}$. Thus, conditioned on (k_1, k_2), c is uniformly distributed over $\mathbb{F}$. Thus, if we define the simulator Sim to sample two random strings k_1, k_2 and a random c, and output $\{(k_i, \mathsf{Enc}_c(0))\}_{i \in [2]}$, it follows from the semantic security of the encryption scheme that the output of the simulator is computationally indistinguishable to the actual messages computed in the protocol.

F.2 Modified Five-Round MPC protocol

We now give a five-round, black-box protocol over point-to-point channels below. This is similar to the protocol given in Section 6 except that we use the special CDS protocol to ensure that malicious parties cannot cheat by sending different messages to different honest parties in the inner protocol. We note that the watchlist protocol is run over point-to-point channels. Specifically, for each round $r \in [4]$, wl_r^i consists of $(\mathsf{wl}_r^{i,1}, \ldots, \mathsf{wl}_r^{i,n})$ where the message $\mathsf{wl}_r^{i,j}$ is the message intended for party P_j.

- **Round-1:** In the first round, the party P_i with input χ_i does the following:
  1. It chooses a random MAC key $k_i \leftarrow \{0,1\}^*$ and sets $z_i := (\chi_i, k_i)$.
  2. It computes $(\phi_1^{i \to 1}, \ldots, \phi_1^{i \to m}) \leftarrow \Phi_1(1^\lambda, i, z_i)$.
  3. It chooses a random subset $K_i \subset [m]$ of size λ and sets $x_{i,j} = K_i$ for every $j \in [n] \setminus \{i\}$.
  4. It chooses a random string $r_{i,h} \leftarrow \{0,1\}^*$ for every $h \in [m]$ and sets $y_{i,j} = \{r_{i,h}, \phi_1^{i \to h}\}_{h \in [m]}$ for every $j \in [n] \setminus \{i\}$.

5. It computes $(\mathsf{wl}_1^{i,1}, \ldots, \mathsf{wl}_1^{i,n}) \leftarrow \mathsf{WL}_1(1^\lambda, i, \{x_{i,j}, y_{i,j}\}_{j \in [n] \setminus \{i\}})$.
6. For each $j \in [n] \setminus \{i\}$, and $r \in [3]$, sample a random string $s_r^{i,j}$ to be used as the shared randomness for the CDS protocol.
7. It sends $\mathsf{wl}_1^{i,j}, \{s_r^{i,j}\}_{r \in [3]}$ to P_j.

- **Round-2:** In the second round, P_i does the following:
  1. For each $h \in [m]$, it computes $\pi_{h,1}^i := \Pi_{h,1}(1^\lambda, i, \phi_1^{i \to h}; r_{i,h})$.
  2. It computes $(\mathsf{wl}_2^{i,1}, \ldots, \mathsf{wl}_2^{i,n}) \leftarrow \mathsf{WL}_2(1^\lambda, i, \{x_{i,j}, y_{i,j}\}_{j \in [n] \setminus \{i\}}, \mathsf{wl}(1))$. (Here, $\mathsf{wl}(r)$ denotes the transcript in the first r rounds of WL.)
  3. It sends $\{\pi_{h,1}^i\}_{h \in [m]}, \mathsf{wl}_2^{i,j}$ to P_j.

- **Round-3:** In the third round, P_i does the following:
  1. For every $h \in [m]$, it computes $\pi_{h,2}^i := \Pi_{h,2}(1^\lambda, i, \phi_1^{i \to h}, \pi_h^i(1); r_{i,h})$. (Here, $\pi_h^i(r)$ denotes the transcript in the first r rounds of Π_h received by P_i.) It additively secret shares $\pi_{h,2}^i$ into n shares $(\pi_{h,2}^{i,1}, \ldots, \pi_{h,2}^{i,n})$.
  2. It computes $(\mathsf{wl}_3^{i,1}, \ldots, \mathsf{wl}_3^{i,n}) \leftarrow \mathsf{WL}_3(1^\lambda, i, \{x_{i,j}, y_{i,j}\}_{j \in [n] \setminus \{i\}}, \mathsf{wl}(2))$.
  3. For each $j \in [n] \setminus \{i\}$, it computes $\mathsf{cds}_{2,1}^{i,j} \leftarrow \mathsf{CDS}_1((\{\pi_h^i(1)\}_{h \in [m]}, \{\pi_{h,2}^{i,j}\}_{h \in [m]}), s_1^{i,j} \oplus s_1^{j,i})$.
  4. It sends $\{\pi_h^i(1)\}_{h \in [m]}, \mathsf{cds}_{2,1}^{i,j}, \mathsf{wl}_3^{i,j}$ to P_j.

- **Round-4:** In the fourth round, P_i does the following:
  1. For each $j, k \in [n]$, it recovers $\{\pi_{h,2}^{j,k}\}_{h \in [m]}$ from the CDS protocol. It uses this to compute $\{\pi_h^i(2)\}_{h \in [m]}$.
  2. For every $h \in [m]$, it computes $\pi_{h,3}^i := \Pi_{h,3}(1^\lambda, i, \phi_1^{i \to h}, \pi_h^i(2); r_{i,h})$. It additively secret shares $\pi_{h,3}^i$ into n shares $(\pi_{h,3}^{i,1}, \ldots, \pi_{h,3}^{i,n})$.
  3. It computes $(\mathsf{wl}_4^{i,1}, \ldots, \mathsf{wl}_4^{i,n}) \leftarrow \mathsf{WL}_4(1^\lambda, i, \{x_{i,j}, y_{i,j}\}_{j \in [n] \setminus \{i\}}, \mathsf{wl}(3))$.
  4. For each $j \in [n] \setminus \{i\}$, it computes $\mathsf{cds}_{3,1}^{i,j} \leftarrow \mathsf{CDS}_1((\{\pi_h^i(2)\}_{h \in [m]}, \{\pi_{h,3}^{i,j}\}_{h \in [m]}), s_2^{i,j} \oplus s_2^{j,i})$.
  5. It sends $\{\pi_h^i(2)\}_{h \in [m]}, \mathsf{cds}_{3,1}^{i,j}, \mathsf{wl}_4^{i,j}$ to P_j.

- **Round-5:** In the fifth round, P_i does the following:
  1. It runs $\mathsf{out}_{\mathsf{WL}}$ on $i, \{x_{i,j}, y_{i,j}\}_{j \in [n] \setminus \{i\}}$, the random tape and $\mathsf{wl}(4)$ to obtain $\{r_{j,h}, \phi_1^{j \to h}\}_{j \in [n] \setminus \{i\}, h \in K_i}$.
  2. For each $j \in [n] \setminus \{i\}$ and $h \in K_i$, it checks:
     (a) If the PRG computations in $\phi_1^{j \to h}$ are correct.
     (b) For each $\ell \in [3]$, whether $\pi_{h,\ell}^j := \Pi_{h,\ell}(1^\lambda, j, \phi_1^{j \to h}, \pi_h(\ell - 1); r_{j,h})$ where $\pi_h(0)$ is set to be the null string.
  3. If any of the above checks fail, then it aborts.
  4. For each $j, k \in [n]$, it recovers $\{\pi_{h,3}^{j,k}\}_{h \in [m]}$ from the CDS protocol. It uses this to compute $\{\pi_h^i(3)\}_{h \in [m]}$.
  5. Else, for each $h \in [m]$, it computes $\pi_{h,4}^i := \Pi_{h,4}(1^\lambda, i, \phi_1^{i \to h}, \pi_h^i(3); r_{i,h})$. It additively secret shares $\pi_{h,4}^i$ into n shares $(\pi_{h,4}^{i,1}, \ldots, \pi_{h,4}^{i,n})$.

6. For each $j \in [n] \setminus \{i\}$, it computes $\mathsf{cds}_{4,1}^{i,j} \leftarrow \mathsf{CDS}_1((\{\pi_h^i(3)\}_{h\in[m]}, \{\pi_{h,4}^{i,j}\}_{h\in[m]}), s_3^{i,j} \oplus s_3^{j,i})$.
7. It sends $\{\pi_h^i(3)\}_{h\in[m]}, \mathsf{cds}_{4,1}^{i,j}$ to P_j.

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

  1. For each $j, k \in [n]$, it recovers $\{\pi_{h,4}^{j,k}\}_{h\in[m]}$ from the CDS protocol. It uses this to compute $\{\pi_h^i(4)\}_{h\in[m]}$.
  2. For every $h \in [m]$, it computes $\phi_2^h := \mathsf{out}_{\Pi_h}(i, \pi_h^i(4))$.
  3. It computes $\mathsf{out}_\Phi(\{\phi_2^h\}_{h\in[m]})$ to recover $(y, \sigma_1, \ldots, \sigma_n)$.
  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.

Proof Sketch. The proof of security of the modified protocol is almost identical to the original protocol. The only difference here is that a corrupted party can additionally send different messages in a particular round of the inner protocol to different honest parties. However, in this case, we rely on the security of the CDS protocol to show that at least one of the shares of the next round message of every honest party is hidden. This ensures that the adversary does not learn any information about the next round message of every honest party.