Round-Optimal Black-Box MPC in the Plain Model

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

Abstract

We give the first construction of a fully black-box round-optimal secure multiparty computation (MPC) protocol in the plain model. Our protocol makes black-box use of a sub-exponentially secure two-message statistical sender private oblivious transfer (SSP-OT), which in turn can be based on (sub-exponential variants of) almost all of the standard cryptographic assumptions known to imply public-key cryptography.

1 Introduction

The exact round complexity of secure computation has been a focus of research in cryptography over the past two decades. This has been especially well-studied in the synchronous setting in the plain model, with up to all-but-one static malicious corruptions. It is known that general-purpose secure multiparty computation (MPC) protocols in this setting admitting a *black-box simulator* require at least 4 rounds of simultaneous exchange [GK96b, KO04, GMPP16].[1] In this work we focus on MPC with black-box simulation. On the positive side, there has been a long sequence of works [GMPP16, BHP17, ACJ17, KS17, BGI$^+$17, BGJ$^+$18, CCG$^+$20] improving the round complexity, culminating in a round-optimal construction that relies on the minimal assumption that a 4-round malicious-secure OT protocol exists [CCG$^+$20].

Black-Box Use of Cryptography. Notably, all MPC protocols discussed above make non-black-box use of cryptography, which is typically associated with significant overheads in efficiency. It is interesting, from both a theoretical and a practical perspective, to realize *fully black-box protocols* [RTV04] where not only does the simulator make black-box use of an adversary, but also the construction itself can be fully specified given just oracle access to the input-output relation of the underlying cryptographic primitives, and without being given any explicit representation of these primitives. In the following, we refer to this standard notion of fully black-box protocols as simply *black-box protocols.* The focus of this work is on the following natural question:

What is the round complexity of black-box MPC in the plain model?

[*]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

[1]By simultaneous message exchange we mean that in each round, every party can send a message over a broadcast channel. However, we allow the adversarial parties to be rushing, meaning that they can wait until they receive all the honest party messages in each round before sending their own messages.

It was only recently that the *concrete* round complexity of black-box MPC in the plain model was studied. Ishai et al. [IKSS21] obtained a *five-round* MPC protocol making only black-box use of a public-key encryption scheme with pseudorandom public keys, along with any 2-message OT protocol satisfying semi-malicious security. They also gave 4-round protocols for a restricted class of functionalities that consist of parallel copies of "sender-receiver" two-party functionalities. While significantly improving over prior works, which required more than 15 rounds, it did not generally match the known 4-round lower bound. Indeed, round-optimal black-box protocols are not known even for the restricted case of two-sided 2PC, where both parties receive the output at the end of the protocol execution. Furthermore, [IKSS21] highlighted significant barriers in extending their techniques to obtain a round-optimal construction.

Our Results. In this work, we overcome these barriers to obtain a 4-round black-box MPC, thereby obtaining the first round-optimal fully-black-box MPC in the plain model for general functions. Our construction makes black-box use of any sub-exponential secure two-message OT, that satisfies a well-studied "statistical sender privacy" (SSP-OT) property. This essentially requires that the sender input remain statistically hidden from an unbounded malicious receiver. Such an OT protocol can be instantiated based on (sub-exponential variants) of standard cryptographic assumptions such as DDH/QR/N^{th} Residuosity/LWE [NP01, AIR01, Kal05, HK12, BD18, DGI$^+$19]. This covers most of the standard cryptographic assumptions known to imply public-key cryptography, with LPN being the most notable exception [2]

On the role of subexponentially secure OT. We stress that even though we rely on subexponentially secure OT, our final simulator still runs in expected polynomial time. This itself may seem counter-intuitive, and indeed we see it as a highlight of our technique and work. Very roughly, the reason why subexponentially secure OT is helpful to us for achieving *polynomial-time* simulation is that we design a protocol that admits *two separate* means for extracting the adversary's input. One is an "optimistic" extraction that runs in expected polynomial time, and the other is a super-polynomial extraction that achieves stronger properties. We use the super-polynomial extraction to essentially "bootstrap" and allow the optimistic extraction to succeed for the purposes of simulation. (See Technical Overview below for more details.) We believe this technique is of independent interest and may inspire progress in other settings where standard polynomial simulation is desired, but there is a need to reduce round complexity beyond a barrier that arises from the need for some component of the protocol to achieve simulation security.

Finally, we note that the 4-round lower bound [GK96b, KO04, GMPP16] holds even when considering protocols that rely on sub-exponential hardness assumptions as long as the simulator runs in (expected) polynomial time.

1.1 Related Work

The black-box round-complexity of general purpose secure computation as well as for specific tasks such as oblivious transfer, zero-knowledge, non-malleable commitments etc., has a long and rich history.

[2]Recently, SSP-OT was constructed from low-noise LPN and a standard derandomization assumption [BF22] (building on [DGH$^+$20]). However, this construction is only secure against quasi-polynomial sized adversaries.

General Purpose MPC. Haitner et al. [HIK$^+$11] gave the first construction of a malicious-secure black-box MPC protocol in the plain model based on any semi-honest secure oblivious transfer. However, the round complexity of this construction grew linearly in the number of parties (denoted by n) even if one starts with a constant round semi-honest OT protocol. A later work of Wee [Wee10] gave a $O(\log^* n)$ black-box protocol by relying on stronger cryptographic assumptions such as dense cryptosystems, or homomorphic encryption, or lossy encryption. This was later improved by Goyal [Goy11] to give a constant round protocol under similar assumptions. Unfortunately, this constant was more than 15 which is a far cry from the lower bound of 4. A recent work of Ishai et al. [IKSS21] gave a black-box five-round protocol based on any PKE with pseudorandom public keys and any two-message OT protocol with semi-malicious security.

Special Secure Computation Tasks. For the case of oblivious transfer, Ostrovsky et al. [ORS15] gave a round-optimal (i.e., a four-round) construction that made black-box use of enhanced trapdoor permutations. Friolo et al. [FMV19] gave a round-optimal black-box construction of OT based on any public key encryption with pseudorandom public keys. Other black-box constructions of four-round OT from lower level primitives were given in [CCG$^+$21, MOSV22].

Ishai et al. [IKSS21] extended these results to the multiparty setting and gave a round-optimal protocol for pairwise oblivious transfer functionality. In the pairwise OT setting, each ordered pair of parties, namely, P_i and P_j execute an OT instance with P_i acting as the sender and P_j acting as the receiver. This can be extended to parallel instances of general two-party sender-receiver functionalities.

Hazay and Venkitasubramanian [HV18] and Khurana et al. [KOS18] gave round-optimal black-box constructions of zero-knowledge arguments based on injective one-way functions. Hazay et al. [HPV20] showed that unless the polynomial hierarchy collapses, all of NP cannot have a black-box zero-knowledge argument based on one-way functions.

Goyal et al. [GLOV12] gave the first constant-round black-box construction of non-malleable commitments based on one-way functions. A latter work of Goyal et al. [GPR16] gave a three-round (which is round-optimal) black-box construction that is secure against a weaker class of synchronizing adversaries assuming the existence of injective one-way functions.

2 Technical Overview

In this section, we give an overview of the main techniques used in our construction of a round-optimal black-box secure multiparty computation protocol.

Starting Point. The starting point of our work is the recent result of Ishai et al. [IKSS21] who gave a construction of a five-round MPC protocol that makes black-box use of any public-key encryption scheme with pseudorandom public keys and any two-message semi-malicious OT protocol.[3] Their protocol is obtained via a round-efficient implementation of the IPS compiler [IPS08] in the plain model.

We note a key component that was used in their instantiation: a four-round black-box protocol that securely implements the *watchlist functionality*. Informally speaking, the watchlist functionality

[3]Recall that semi-malicious adversaries are stronger than the standard semi-honest adversaries and are allowed to fix the random tape of adversarial parties to arbitrary values. However, like in the semi-honest setting, they are forced to follow the protocol specification.

is an n-party functionality where each ordered pair of parties (P_i, P_j) where $i, j \in [n]$ are involved in a k-out-of-m OT instance with P_i acting as the sender and P_j acting as the receiver. Using this four-round watchlist protocol, Ishai et al. [IKSS21] showed that with an additional round of interaction, it is possible to securely compute any multiparty functionality. Furthermore, the resulting protocol only made black-box use of cryptographic primitives.

Going Below Five Rounds. In the same work, Ishai et al. [IKSS21] also observed that to get a four-round protocol (which is round-optimal) in the plain model by making use of the IPS compiler, one needs a three-round watchlist protocol. However, such a protocol cannot satisfy the standard simulation based security definition w.r.t. a simulator that only makes black-box use of the adversary. This is because such a simulation-secure watchlist protocol almost directly implies a three-round protocol for oblivious transfer that satisfies standard simulation security. We know that such a protocol is impossible to construct (even with non-black-box use of cryptography) if the simulator uses the adversary in a black-box manner [KO04]. Furthermore, to make matters more complicated, the proof of security of the overall compiler given in [IKSS21] crucially relied on the watchlist protocol to satisfy the standard simulation-style definition. Therefore, to go below five rounds and obtain a round-optimal construction, we need to come up with a new set of techniques.

Our Approach in a Nutshell. In this work, we show how to instantiate the IPS compiler using a weaker notion of watchlists, that we call *watchlists with promise security.* As one of our main contributions, we give a construction of a three-round watchlist protocol that satisfies promise security. In Section 2.1, we motivate the definition of this weaker watchlist protocol and show how it can be used to instantiate the IPS compiler and in Section 2.2, we give the main ideas in constructing such a watchlist protocol.

2.1 Instantiating the IPS Compiler with Three-Round Watchlist

What Security can be achieved in Three Rounds? The work of Ishai et al. [IKSS21] gave a round-preserving compiler that transforms any two-party computation protocol that satisfies certain additional properties (which we will ignore for the moment) to a watchlist protocol. To understand what security properties can be achieved by a three-round watchlist protocol, let us first try to understand what type of security can be achieved by a three-round 2PC protocol.

Recall that in the standard two-party protocol setting, there is a receiver who holds an input x and there is a sender who holds an input y. At the end of the protocol, the receiver obtains the output of $f(x, y)$ for some pre-determined functionality f. If we consider three-round protocols for the above task, then the first and the third round messages in the protocol are sent by the sender and the second round message is sent by the receiver.[4] As the sender is tasked with sending both the first and the third round message, a simulator could potentially rewind the second and the third round messages in the protocol and extract the effective private input from an adversarial sender. In other words, a three-round 2PC protocol could satisfy standard simulation-based security definition against malicious senders. However, the receiver in this protocol is only sending a single message, namely, the second round message. In fact, it is impossible to design a black-box PPT simulator that could extract the effective private input from an adversarial receiver.

[4]We note that any protocol, even one in the bidirectional communication model, can be reduced to this setting.

The key observation is that if we allow the simulator against malicious receivers to run in *super-polynomial* time, then such a simulator can extract the effective receiver input and provide security against malicious receivers. Therefore, in the three-round setting, we can hope to construct a two-party protocol that satisfies standard simulation based security against malicious senders and super-polynomial time simulation security against malicious receivers. Indeed, as we explain later, we give a construction of such a three-round protocol that makes black-box use of a sub-exponentially hard two-message OT protocol with statistical sender security. Such an OT protocol is known from the (sub-exponential variant) of standard cryptographic hardness assumptions such as DDH/N^{th} residuosity/LWE/QR [AIR01, NP01, Kal05, HK12, BD18, DGI$^+$19].

Instantiating the IPS Compiler with the Three-Round Watchlist. Given the two-party protocol above, we could hope to obtain a three-round watchlist satisfying "semi-SPS" security by following ideas in prior work [IKSS21]. If this were possible, could we directly get a four-round MPC protocol by instantiating the IPS compiler with this "semi-SPS" three-round watchlist protocol? Unfortunately, this is not quite possible, as we now explain. To understand this better, we give a brief overview of the IPS compiler which is simplified and tailored to constructing a four-round protocol. The IPS compiler makes use of the following components:

- A two-round client-server MPC protocol that is secure against a malicious adversary that corrupts an arbitrary number of clients and a constant fraction of the servers. This is called as the outer protocol. Such an outer protocol was constructed by Ishai et al. [IKP10, Pas12] by making black-box use of any PRG.

- A four-round inner protocol that satisfies the following robustness property. Specifically, even if the adversary behaves maliciously and deviates arbitrarily from the protocol specification in the first three rounds, it cannot learn any information about the inputs of the honest parties. Furthermore, if the adversary is able to produce an input, random tape that correctly explains that the messages sent by it in the first three rounds, then the last round message from the honest parties only reveals the output of the functionality.[5]

- A three-round watchlist protocol that satisfies the standard extraction of the adversarial sender inputs and super-polynomial time extraction of the adversarial receiver inputs.

In the compiled protocol, each party plays the role of a client in the outer protocol and computation done by the servers are emulated by the inner protocol. To ensure that that the adversary only cheats in at most a small number of these inner protocol executions, we make use of the *watchlist protocol*. Specifically, each party acting as the receiver in the watchlist protocol chooses a random subset of k executions as part of its secret watchlist. Every other party acting as the sender uses the input, randomness used in each of the inner protocol executions as the sender inputs. This watchlist protocol is run in parallel with the first three rounds of the inner protocol. At the end of the third round, each party checks if the input, randomness pair provided by every other party corresponding to its watched executions are consistent. If it detects any inconsistency, then it aborts. Using standard probabilistic arguments, it is possible to show that if the honest parties have not aborted

[5]For technical reasons, we actually need the inner protocol to run in three rounds instead of four rounds. However, to keep the exposition simple, we will ignore this in the overview. In the main body, we give a black-box construction of such a three-round inner protocol based on two-round semi-malicious OT protocol (which is implied by two-round SSP OT). This construction builds on the protocols given in [GS18, PS21].

at the end of their watchlist check, then the adversary only deviates in a tiny constant fraction of the inner protocol executions. These deviations can be directly mapped to the corresponding server corruptions in the outer protocol. Since the outer protocol is secure against a constant fraction of the server corruptions, security of the overall protocol follows.

While the above intuition seems sound, we encounter a major issue while formalizing it. In particular, recall that we are aiming for standard polynomial security for our 4-round protocol, but we are relying on super-polynomial time extraction as an ingredient. Thus, we are only able to show that this protocol satisfies security via a super-polynomial time simulator. The "super-polynomial" part in this simulator is needed to extract the receiver inputs used by the adversarial parties in the watchlist protocol. Recall that in the watchlist protocol, the adversarial receiver inputs correspond to the set of watched executions of the corrupted parties. We need to extract this information in order to invoke the security of the outer protocol.[6] Further, the simulator also needs to additionally extract the adversarial sender inputs. As mentioned earlier, we cannot hope to simultaneously achieve efficient polynomial time extraction of both the sender and the receiver inputs.

Our Solution: "Promise-Style" Extraction. In order to get around this issue, we use a "promise-style" extraction technique that is inspired by the notion of Promise Zero-Knowledge [BGJ$^+$18]. Specifically, we seek to devise an alternative polynomial-time extraction system that guarantees extraction of the adversarial receiver inputs only against those adversaries that send a valid third round message in the watchlist protocol (with non-negligible probability). For all other adversaries, we do not provide any guarantees. Let us now explain how this weaker extraction guarantee is sufficient to instantiate the IPS compiler.

The simulator of the compiled protocol starts generating the first-round messages of the outer protocol using some default inputs for the honest parties. Note that these first round messages correspond to the inputs to the inner protocol executions. The simulator uses these "dummy" inputs to the inner protocol and starts interacting with the adversary for the first three rounds. If the adversary aborts during this interaction, or fails to send a valid third round message in the watchlist protocol, then the simulator simply outputs the view of this adversary. On the other hand, if the adversary sends a valid third round message in the watchlist protocol, then the simulator uses the "promise-style" extractor to extract the set of watched executions. This information is then used by the simulator to simulate the messages in the main thread (using Goldreich-Kahan simulation technique [GK96a]).

A subtle point to note here is that the third round message in the watchlist protocol is sent by the adversary only after it receives the third round message from the honest parties (as we are considering rushing adversaries). However, the third round message of the watchlist protocol delivers the input, randomness used by the honest parties corresponding to the adversarial watched executions. Recall that the simulator described above uses "dummy" inputs in the inner protocol executions and tries to extract the adversarial watched executions. This will succeed only if the distribution of the messages generated by the simulator is computationally indistinguishable to the messages in the real protocol. Specifically, to prove this indistinguishability, we need to make sure that the output of the watchlist protocol when using the real inputs is indistinguishable to the case when the simulator uses default inputs.

[6]Specifically, the set of watched executions of the adversarial parties correspond to a subset of the corrupted servers in the outer protocol. To invoke the security of the outer protocol, we need to extract this information from the watchlist messages.

To argue this, we rely on the security of the outer protocol. Recall that the inputs given to the inner protocol executions correspond to the messages sent by the clients to the servers. By corrupting the servers corresponding to the adversarial watched executions, we are guaranteed that the first round message sent to these servers reveals no information about the inputs of the honest clients. To give a bit more details, this is realized by first relying on the SPS security of the watchlist protocol against adversarial receivers to extract the adversarial watched executions, and then switch the input to a default value by relying the security of the outer protocol, and then switch back to an honest watchlist execution using the default inputs.

Another point to note here is that we cannot guarantee perfect extraction of the adversarial receiver inputs even if it sends a valid third round message with non-negligible probability. Due to technical reasons, we can only guarantee "almost" perfect extraction. By this, we mean that whenever the output received by the adversarial receiver is not $\perp$, the output of the promise extractor is identical to the SPS extractor. In other cases, there are no guarantees about the extracted value. We show that this weaker property is sufficient to instantiate the IPS compiler. Roughly, this is because if the output of the watchlist protocol is provided to the adversary is $\perp$, the adversary learns no information about the input, randomness for any inner protocol execution. Hence, if the promise extractor "over-extracts" the adversarial watched executions, this does not create any trouble with the simulation.

2.2 Constructing Three-Round Watchlists with Promise Extraction

A core ingredient of our black-box MPC protocol is a three-round "watchlist" protocol with promise-style extraction guarantees. For every $i \in [n], j \in [n] \setminus \{i\}$, this functionality enables P_i to choose a (private) subset $K \subseteq [m]$ of protocol executions of size k, and obtain the input and randomness used by P_j in all executions in the set K, while all other input and randomness values of P_j remain hidden from P_i.

Our first goal is to develop a three round protocol that realizes the watchlist functionality in the plain model in the presence of *malicious* corruptions, with super-polynomial simulation and (polynomial) promise-style extraction. Following [IKSS21], we observe that it would suffice to implement "sender non-malleable" OTs with super-polynomial simulation-based "real/ideal" security and with promise-style polynomial extraction; where in the (i, j)-th execution for $i \in [n], j \in [n] \setminus \{i\}$, P_i is the receiver and P_j is the sender. P_j's input to the OT will be the input and randomness it used in each of the m instances of the inner protocol, and P_i's input is a random subset K of $[m]$ of size k. By sender non-malleability, we mean that the adversarial parties cannot maul the sender messages in an OT execution with an honest party to obtain a "related" sender inputs in an OT execution with an honest receiver.

The work of [IKSS21] showed how to implement such sender non-malleable OT in four rounds from any *four-round simulation-secure two-party computation protocol* with certain additional properties (which we ignore for the momemt). Since we need three-round watchlists, we would need to begin with three-round two-party computation, which is impossible to realize with black-box polynomial-time simulation security. Nevertheless, we show that it is possible to realize such two-party computation with super-polynomial simulation and promise-style extraction, which is one of our key technical contributions. We describe this in the next subsection; here we discuss how such a two-party protocol can be compiled into 3-round non-malleable OT.

Our overall approach builds on [IKSS21], but also diverges in some key technical aspects. Like [IKSS21], our construction relies on a secure two-party protocol between a sender and a re-

Sender Inputs: $\mathsf{m}_0, \mathsf{L}_0, \mathsf{M}_0, \mathsf{R}_0, \mathsf{m}_1, \mathsf{L}_1, \mathsf{M}_1, \mathsf{R}_1$, **Receiver Inputs:** b, c
The functionality $\mathcal{F}$ is defined as follows.

1. Check if $\mathsf{L}_0, \mathsf{M}_0, \mathsf{R}_0$ is a valid encoding of m_0 and if not, output $\perp$.
2. Check if $\mathsf{L}_1, \mathsf{M}_1, \mathsf{R}_1$ is a valid encoding of m_1 and if not, output $\perp$.
3. If $c = 0$, output $(\mathsf{m}_b, \mathsf{L}_0, \mathsf{L}_1)$.
4. If $c = 1$, output $(\mathsf{m}_b, \mathsf{M}_0, \mathsf{M}_1)$.
5. If $c = 2$, output $(\mathsf{m}_b, \mathsf{R}_0, \mathsf{R}_1)$.

Figure 1: The functionality $\mathcal{F}$

ceiver realizing a special functionality $\mathcal{F}$ (described in Figure 1). Unlike [IKSS21], we must develop a *three-round* compiler instead of a four round one.

In the [IKSS21] compiler, the sender S on input $(\mathsf{m}_0, \mathsf{m}_1)$ first encodes these messages using an appropriate 2-split-state non-malleable code $(\mathsf{Enc}, \mathsf{Dec})$.[7] For technical reasons pertaining to the use of watchlists in our final protocol, we require our watchlists to satisfy 1-*rewinding security*, i.e., no adversary should be able to distinguish *the joint distribution* of a main and a rewinding thread (with common prefix) from the real distribution, from those sampled according to the simulated distribution. This was not needed by [IKSS21], but this requirement in our setting necessitates deviating from the [IKSS21] template, relying on (a special type of) 3-split-state non-malleable code – specifically one that is also a 3-out-of-3 secret sharing scheme – instead of 2-split-state non-malleable codes.

Specifically, our sender encodes m_0 into $\mathsf{L}_0, \mathsf{M}_0, \mathsf{R}_0$ and encodes m_1 into $\mathsf{L}_1, \mathsf{M}_1, \mathsf{R}_1$. The receiver obtains input a choice bit $b \in \{0, 1\}$, and additionally samples a uniformly random $c \in \{0, 1, 2\}$. S and $\mathcal{R}$ invoke a two-party secure protocol Π to compute functionality $\mathcal{F}$, described in Figure 1.

We note that the ideal functionality $\mathcal{F}$ only reveals m_b to the receiver, and statistically hides m_{1-b}. This is because the receiver obtains *only one of* L_{1-b}, M_{1-b} and $\mathcal{R}_{1-b}$, and secrecy follows from the security of the secret sharing scheme. Thus, given one of the states the message m_{1-b} is information-theoretically hidden. Further, even given two executions of the ideal functionality on the *same sender inputs*, same receiver input b, and different receiver challenges c, the receiver only obtains m_b and two out of L_{1-b}, M_{1-b} and $\mathcal{R}_{1-b}$. Given two out of these three shares, m_{1-b} is again statistically hidden. Indeed, when $\mathcal{F}$ is realized via a secure protocol Π, m_{1-b} continues to be computationally hidden even given a main and rewinding thread (with same inputs $\mathsf{m}_0, \mathsf{m}_1, b$). This protocol Π makes only black-box use of cryptography, and can be based on black-box access to our three-round two-party computation protocol that additionally satisfies certain amount of rewinding security, which we discuss in the next subsection.

[7] Recall that a split-state non-malleable code $(\mathsf{Enc}, \mathsf{Dec})$ encodes any message m into multiple states, such that the distribution of the tampered message obtained by tampering the each state individually is independent of m.

Proving Sender Non-Malleability. We must prove that running this protocol Π between every pair of parties in parallel securely realizes the watchlist functionality. We model the adversary as a man-in-the-middle, which acts as a receiver in "left" sessions and as sender in "right" sessions. We require that there is a simulator-extractor $\mathsf{Sim\text{-}Ext}$ that given the inputs of all honest receivers (in all right sessions), is able to extract all the implicit inputs used by the man-in-the-middle in all its right sessions. Crucially, $\mathsf{Sim\text{-}Ext}$ *does not* have access to the inputs of honest senders. Since the underlying protocol Π may be susceptible to arbitrary mauling attacks, achieving this property is non-trivial, as we discuss next.

Similar to [IKSS21], we use the specific way that sender inputs are encoded to introduce an *alternate* extraction mechanism. Specifically, one could imagine rewinding the second and the third round message of Π twice, with first round message fixed, and using inputs $c = 0$, $c = 1$ and $c = 2$ on behalf of the honest receiver in the real and rewinding threads, respectively. Our two-party computation protocol will be developed in such a way that fixing the first round message will fix all other inputs $\mathsf{m}_0, \mathsf{m}_1, b$ in all left and right sessions. Let us make the simplifying assumption that our adversary does not abort. Therefore, we expect to obtain outputs $(\widetilde{\mathsf{L}}_0, \widetilde{\mathsf{L}}_1)$, $(\widetilde{\mathsf{M}}_0, \widetilde{\mathsf{M}}_1)$ 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.

The Need for 2-Rewinding Security. Before we can rely on non-malleable codes to formally argue security, we need to replace the two-party protocol Π with its *simulated version.* At the same time, we need to argue that the joint distribution of values extracted from the strategy above (via extracting $(\widetilde{\mathsf{L}}_0, \widetilde{\mathsf{L}}_1)$, $(\widetilde{\mathsf{M}}_0, \widetilde{\mathsf{M}}_1)$ and $(\widetilde{\mathsf{R}}_0, \widetilde{\mathsf{R}}_1)$) from the simulated two-party protocol, matches the distribution in the real protocol. This requires the two-party protocol Π to satisfy a stronger security property, that we call 2-rewind sender security. This roughly means that any adversarial receiver/MIM that rewinds the honest sender one time in the third and fourth rounds, with its input $\widetilde{c}$ set to a possibly different value, does not learn more than the output of $\mathcal{F}$ on (fixed) inputs $(\mathsf{m}_0, \mathsf{m}_1, \mathsf{L}_0, \mathsf{L}_1, \mathsf{M}_0, \mathsf{M}_1, \mathsf{R}_0, \mathsf{R}_1, \widetilde{b}, \widetilde{c} = 0)$, $(\mathsf{m}_0, \mathsf{m}_1, \mathsf{L}_0, \mathsf{L}_1, \mathsf{R}_0, \mathsf{R}_1, \widetilde{b}, \widetilde{c} = 1)$ and $(\mathsf{m}_0, \mathsf{m}_1, \mathsf{L}_0, \mathsf{L}_1, \mathsf{R}_0, \mathsf{R}_1, \widetilde{b}, \widetilde{c} = 2)$. This can be formalized by demonstrating the existence of a simulator that simulates the receiver's view in the real and rewinding threads, given only $(\mathsf{m}_{\widetilde{\mathsf{b}}}, \mathsf{L}_0, \mathsf{L}_1)$ in the main thread, and $(\mathsf{m}_{\widetilde{\mathsf{b}}}, \mathsf{M}_0, \mathsf{M}_1)$, $(\mathsf{m}_{\widetilde{\mathsf{b}}}, \mathsf{R}_0, \mathsf{R}_1)$ respectively in each of the rewinding threads (w.l.o.g.). Now, it may seem like the sum total of this information could essentially allow the receiver to recover $\mathsf{m}_{1-\widetilde{\mathsf{b}}}$. Yet, we show that if Π satisfies this property, it becomes possible to replace $\mathsf{m}_{1-\widetilde{\mathsf{b}}}$ with an arbitrary value (say 0^λ). Here we make use of the fact that the different states of the non-malleable code are available to the MIM in separate (i.e. real and rewinding) executions, which allows us to rely on the security guarantees provided by non-malleable codes, by arguing that each of these states are essentially tampered by *independent* functions.

Finally, we note that just as in [IKSS21], we require these codes to satisfy *many-many* non-malleability. At a high level, these are codes that are secure against multiple tamperings of a codeword [CGL16]. We note that [GSZ21] construction of 3-out-of-3 non-malleable secret sharing satisfies all the required properties (if instantiated with the CGL non-malleable code). Also following [IKSS21], to deal with adversaries who might abort, we will modify the protocol and functionality $\mathcal{F}$ so that instead of encoding (m_0, m_1) a single time, the sender generates λ (where λ is the security parameter) fresh encodings $\{(\mathsf{L}^i_\mathsf{b}, \mathsf{M}^i_\mathsf{b}, \mathsf{R}^i_\mathsf{b})\}_{i\in[\lambda],\mathsf{b}\in\{0,1\}}$ 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 $i \in [n], \mathsf{b} \in \{0, 1\}$,

$\{(\mathsf{L}^i_\mathsf{b}, \mathsf{M}^i_\mathsf{b}, \mathsf{R}^i_\mathsf{b})\}_{i\in[\lambda],\mathsf{b}\in\{0,1\}}$ encode m_b. If the check fails, $\mathcal{F}$ outputs $\bot$. If it passes, then for every $i \in [n]$, it outputs $(\mathsf{L}^i_0, \mathsf{L}^i_1)$ if $c_i = 0$, $(\mathsf{M}^i_0, \mathsf{M}^i_1)$ if $c_i = 1$, and otherwise, outputs $(\mathsf{R}^i_0, \mathsf{R}^i_1)$. We also recall that our watchlists need to satisfy super-polynomial simulation with "promise-style" extraction, but we note that these properties in fact carry over from the underlying special two-party computation protocol.

2.3 Constructing Three-Round 2PC with Special Extraction

In this subsection, we explain the key ideas behind our construction of a three-round 2PC that satisfies the "promise-style" extraction guarantee and "2-rewinding" sender security.

3-Round OT Protocol. As a first step, we construct a three-round black-box OT protocol that satisfies standard simulation-based security against malicious senders and super-polynomial time simulation security against malicious receivers. For this purpose, we rely on a (sub-exponentially hard) two-round OT protocol that has super-polynomial time simulation security against malicious receivers. To enable polynomial time extraction of the malicious sender input, we additionally require the sender to generate an extractable commitment to its input. To ensure the consistency of inputs used in the extractable commitment and the ones used in the OT protocol, we rely on the IPS compiler. Specifically, we use the 1-out-of-2 SPS OT to construct a k-out-of-m SPS OT protocol (using Yao's garbled circuits) and use this as the watchlist protocol. We show that this watchlist protocol is sufficient to instantiate the IPS compiler when we only require SPS security against malicious receivers. The formal description of the construction and the proof of security appears in Section 4.

3-Round 2PC. As a next step, we use the above OT protocol to construct a three-round 2PC protocol that satisfies standard simulation security against malicious senders and SPS security against malicious receivers. This step involves standard tools and closely follows the construction given in [IKSS21]. Additionally, we also show how to add 2-rewinding sender security to this protocol. Specifically, we show that if the underlying 3-round OT is 2-rewinding sender secure and we also have a 2-rewinding secure extractable commitment scheme (which was constructed in [BGJ$^+$18]), we get a 2-rewinding sender secure 2PC protocol. Further, we note that 2-rewinding sender security of our 3-round OT protocol just boils down to instantiating the underling extractable commitment on the sender side (as explained earlier) with a 2-rewinding secure one, and we instantiate this with the construction given in [BGJ$^+$18]. The formal description of the construction and the proof of security appears in Section 5.

3-Round 2PC with Special Extraction. We then use the above 3-round 2PC protocol to construct a protocol that additionally satisfies the "promise-style" extraction guarantee. To achieve this, we require the receiver to commit to its input (as well as the randomness) used in the 2PC protocol via a three-round extractable commitment. Again, as in the case of OT protocol, we need to make sure that the inputs committed via the extractable commitment is consistent with the inputs used in the 2PC protocol. As before, we rely on the IPS compiler but we observe that we do not need the "full-blown" watchlist protocol. Instead, we require the sender in the second round to send a set of executions to be opened in the *clear* and the receiver in the final round opens the extractable commitment corresponding to these executions. The sender then checks whether the input, randomness committed via the extractable commitment is consistent with the messages sent in the 2PC protocol. If they are consistent for randomly opened set of executions, then by standard statistical argument, we can show that they are consistent for a majority of the executions with overwhelming probability. This allows us to rewind and extract the receiver's input via the extractable commitment. We note that we are only able to guarantee "almost" perfect extraction due to the existence of a "small" set of inconsistent executions. Specifically, the "small" set of inconsistent executions could force the output of the watchlist protocol to be $\perp$, but even in this case, our polynomial time extract could extract some receiver input. But as explained earlier, this is not problematic and is sufficient to instantiate the IPS compiler. We also note that if the underlying 2PC protocol is 2-rewinding sender secure then this property is inherited by the 2PC protocol with special extraction as well. The construction and proof of security appears in Section 6.

2.4 Organization

In Section 3, we give the formal definitions of the all the building blocks needed in our protocol constructions. In Section 4, we give our construction of a 3-round oblivious transfer protocol. In Section 5, we give our construction of 3-round 2PC protocol. In Section 6, we show how to add promise-style extraction guarantees for the receiver. In Section 7, we give our construction of the watchlist protocol that satisfies promise-style extraction. In Section 8, we give our construction of a three-round inner protocol. Finally, in Section 9, we give our 4-round fully black-box MPC protocol.

3 Preliminaries

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

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

Definition 3.1. *We say that two distribution ensembles $\{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $\{Y_\lambda\}_{\lambda\in\mathbb{N}}$ are (T,ϵ)-indistinguishable if for every non-uniform distinguisher D running in time $T(\lambda)$ (abbreviated as T), we have $|\Pr[D(1^\lambda, X_\lambda) = 1]| - \Pr[D(1^\lambda, Y_\lambda) = 1]| \leq \epsilon(\lambda)$.*

Two-Message OT with Super-Polynomial Time Sender Security. We give the definition of a 1-out-of-2 OT protocol that satisfies super-polynomial simulation security against malicious receivers and indistinguishability-based security against senders.

Definition 3.2. *A tuple of algorithms* $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{out}_{\mathsf{OT}})$ *implementing the* 1*-out-of-*2 *oblivious transfer functionality has super-polynomial simulation security against malicious receivers and indistinguishability-based security against senders if:*

- ***Correctness.*** *For every input* $b \in \{0,1\}$ *of the receiver and any input* (s_0, s_1) *of the sender and for any choice of random tape* r *of receiver, we have:*

$$\Pr[\mathsf{out}_{\mathsf{OT}}(\mathsf{otm}_1, \mathsf{otm}_2, (b, r)) = s_b] = 1$$

where $\mathsf{otm}_1 := \mathsf{OT}_1(1^\lambda, b; r)$ *and* $\mathsf{otm}_2 \leftarrow \mathsf{OT}_2(\mathsf{otm}_1, (s_0, s_1))$.

- ***Super-Polynomial Simulation Security against Malicious Receivers.*** *There exits a super-polynomial time algorithm* Ext *that runs in time* $T_1(\lambda)$ *(abbreviated as* T_1*) such that for any non-uniform PPT* $\mathcal{A}$ *and for any sender's input* (s_0, s_1)*, we have:*

$$\Big\{\mathsf{OT}_2(\mathsf{otm}_1, (s_0, s_1)) : \mathsf{otm}_1 \leftarrow \mathcal{A}(1^\lambda)\Big\} \approx_c \\ \Big\{\mathsf{OT}_2(\mathsf{otm}_1, (s_b, s_b)) : \mathsf{otm}_1 \leftarrow \mathcal{A}(1^\lambda), b \leftarrow \mathsf{Ext}(\mathsf{otm}_1)\Big\}$$

- ***Indistinguishability against Malicious Senders.*** *For any PPT adversary* $\mathcal{A}$ *corrupting the sender* S*, we have:*

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

Two-message OT protocols satisfying the above definition[8] is known from standard cryptographic assumptions such as DDH/LWE/QR/N^{th} residuosity [AIR01, NP01, BD18, DGI+19, Kal05, HK12] and additionally from (low-noise) LPN together with a standard derandomization assumption [BF22]. These protocols satisfy a stronger security property wherein the two distributions described in the SPS security against malicious receivers are statistically close.

We also consider generalizations of the above definition where we require (T, ϵ)-indistinguishability against malicious senders. Under the assumption that the above mentioned problems are (T, ϵ)-secure, we get construction of a two-message OT protocol that satisfies (T, ϵ)-indistinguishability against malicious senders.

Two-Round OT protocol with Equivocal Receiver Security. Below, we recall the definition of a two-round OT protocol that has equivocal receiver security from [IKSS21].

Definition 3.3. *A two-round OT protocol is said to be equivocal receiver secure if:*

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

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

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

[8]In fact, these constructions satisfy a stronger form of definition where the malicious receiver could be unbounded.

- ***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* $b \in \{0,1\}$*,*

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

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

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

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

[IKSS21] gave a construction of such a protocol based on black-box access to a two-round semi-malicious OT protocol. Then, there exists a two-round OT protocol satisfying Definition 3.3.

Theorem 3.4. *Assume black-box access to a two-round OT protocol that is secure against semi-malicious adversaries.*

k-Rewinding Security for Extractable Commitments. We define the notion of k-rewinding security for an extractable commitment where k is some constant. Consider the following experiment between a committer $\mathcal{C}$ and any (possibly cheating) receiver $\mathcal{R}^*$.

- **Experiment** $\mathcal{E}(m)$**:**
  - $\mathcal{R}^*$ interacts with $\mathcal{C}$ who obtains input m. The complete one execution of the protocol ECom. $\mathcal{R}^*$ receives values (e_1, e_3) in rounds 1 and 3 respectively.
  - Then, following is repeated k times. $\mathcal{R}^*$ rewinds $\mathcal{C}$ to the beginning of round 2. $\mathcal{R}^*$ sends $\mathcal{C}$ a new second round message e_2^* and receives a message e_3^* in the third round.
  - At the end of the experiment, $\mathcal{R}^*$ outputs $0/1$.

Definition 3.5 (k-Rewinding Security)**.** *An extractable commitment scheme denoted by* $\mathsf{ECom} = (\mathsf{ECom}_1, \mathsf{ECom}_2, \mathsf{ECom}_3, \mathsf{ECom}_{\mathsf{Out}}, \mathsf{ECom}_{\mathsf{Valid}})$ *achieves k-rewinding security if, for every non-uniform PPT receiver* $\mathcal{R}^*$ *in the above experiment* E *and every pair of messages* m_0, m_1*, there exists a negligible function* $\mu(\cdot)$ *such that:*

$$|\Pr[\mathcal{R}^* = 1 | \mathcal{E}(m_0)] - \Pr[\mathcal{R}^* = 1 | \mathcal{E}(m_1)]| \leq \mu(\lambda)$$

where the probability is over the random coins of $\mathcal{C}$*.*

We can generalize the above definition to provide (T, ϵ)-security against malicious receivers that run in time $T(\lambda)$. Badrinarayanan et al. [BGJ+18] gave a construction of a k-rewinding secure extractable commitment that makes black-box use of a non-interactive commitment scheme. If the commitment scheme is (T, ϵ)-hiding, then their construction provides (T, ϵ) security.

Garbled Circuits Below we recall the definition of garbling scheme for circuits [Yao86] (see Applebaum et al. [AIK04, AIK05], Lindell and Pinkas [LP09] and Bellare et al. [BHR12] for a detailed proof and further discussion).

Definition 3.6. *A garbling scheme for circuits is a tuple of PPT algorithms* $(\mathsf{Garble}, \mathsf{Eval})$*.* Garble *is the circuit garbling procedure and* Eval *is the corresponding evaluation procedure. More formally:*

- $\widetilde{C} \leftarrow \mathsf{Garble}\left(1^\lambda, C, \{\mathsf{lab}_{w,b}\}_{w\in\mathsf{inp}, b\in\{0,1\}}\right)$*:* Garble *takes as input a security parameter* 1^λ*, a circuit* C *along with labels* $\mathsf{lab}_{w,b}$ *where* $w \in \mathsf{inp}$ *(*inp *is the set of input wires of* C*) and* $b \in \{0,1\}$ *and outputs a* garbled circuit $\widetilde{C}$*. Each label* $\mathsf{lab}_{w,b}$ *is assumed to be in* $\{0,1\}^\lambda$*.*

- $y \leftarrow \mathsf{Eval}\left(\widetilde{C}, \{\mathsf{lab}_{w,x_w}\}_{w\in\mathsf{inp}}\right)$*: Given a garbled circuit* $\widetilde{C}$ *and a sequence of input labels* $\{\mathsf{lab}_{w,x_w}\}_{w\in\mathsf{inp}}$ *(referred to as the garbled input),* Eval *outputs a string* y*.*

We require it to satisfy the following two properties.

- ***Correctness.*** *For correctness, we require that for any circuit* C*, input* $x \in \{0,1\}^{|\mathsf{inp}|}$ *and for any set of wire labels* $\{\mathsf{lab}_{w,b}\}_{w\in\mathsf{inp}, b\in\{0,1\}}$ *we have that:*

$$\Pr\left[C(x) = \mathsf{Eval}\left(\widetilde{C}, \{\mathsf{lab}_{w,x_w}\}_{w\in\mathsf{inp}}\right)\right] = 1$$

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

- ***Security.*** *For security, we require that there exists a PPT simulator* Sim *such that for any polynomial-time generated circuit family* $C = C(\lambda)$ *and input* $x \in \{0,1\}^{|\mathsf{inp}|}$ *for* C*, we have that*

$$\left(\widetilde{C}, \{\mathsf{lab}_{w,x_w}\}_{w\in\mathsf{inp}}\right) \approx_c \mathsf{Sim}\left(1^\lambda, 1^{|C|}, 1^{|x|}, C(x)\right)$$

  where $\widetilde{C} \leftarrow \mathsf{Garble}\left(1^\lambda, C, \{\mathsf{lab}_{w,b}\}_{w\in\mathsf{inp}, b\in\{0,1\}}\right)$ *and* $\{\mathsf{lab}_{w,b}\}_{w\in\mathsf{inp}, b\in\{0,1\}}$ *is sampled uniformly.*

MPC Definition. We recall the standard simulation-based security definition of an MPC protocol in Appendix B.

Split-State 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 3.7 (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*

- $\mathsf{NM.Code}(m) \to (L, M, R)$ *where* $L \in \mathcal{L}$*,* $M \in \mathcal{M}$ *and* $R \in \mathcal{R}$ *(we will assume that* $\mathcal{L} = \mathcal{M} = \mathcal{R}$*) are a three-out-of-three secret sharing of the message,*

- *For every* $m \in \{0,1\}^{p(\lambda)}$*,*

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

- *For every set of functions* $f = (f_1, f_2, \dots f_{\ell(\lambda)}), g = (g_1, g_2, \dots g_{\ell(\lambda)}), h = (h_1, h_2, \dots h_{\ell(\lambda)})$ *and every set of permutations* $\{\sigma_i\}_{i\in[\ell(\lambda)]}, \sigma'$ *on* (L, M, R) *there exists a random variable* $\mathcal{D}_{f,g,h,\sigma,\sigma'}$ *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 the statistical distance between the distributions*

$$\sigma'(\mathsf{L}), \sigma'(\mathsf{M}), \{\mathsf{NM.Decode}\big(f_i(\sigma_i(\mathsf{L})), g_i(\sigma_i(\mathsf{M})), h_i(\sigma_i(\mathsf{R}))\big)\}_{i\in[\ell(\lambda)]}$$

$$\text{and } \big(\mathsf{replace}(\mathcal{D}_{f,g,h,\sigma,\sigma'}, m)\big) \text{ where } (\mathsf{L}, \mathsf{M}, \mathsf{R} \leftarrow \mathsf{NM.Code}(m))$$

  is at most $\epsilon(\lambda)$, *where the function* $\mathsf{replace} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ *replaces all occurrences of* same^* *in its first input with its second input, and outputs the result.*

We note that the construction of non-malleable secret sharing in [GSZ20] can be proven to satisfy this definition. This is already implicit in [GSZ20] for the case of single tampering but extension of their proof to the case of multiple tamperings follows directly if we use a strong two-source non-malleable extractors that is multi-tamperable [CGL16].Thus, we have the following:

Lemma 3.8. *[GSZ20] 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 3.7 with efficient encoding and decoding algorithms with code length* $q(\lambda)$, *rate* $q(\lambda)^{-\Omega(1)}$ *and error* $2^{-q(\lambda)^{\Omega(1)}}$.

Low-Depth Proofs 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 $\mathsf{NC1}$. We refer to the transformation as a low-depth proof, and we require such a proof to satisfy the following definition.

Definition 3.9 (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* NC^1.

It is shown in [IKSS21] building on [GGH+13] how such a non-interactive proof can be constructed in a simple way. Looking ahead, our construction of watchlists makes use of a (malleable) two-party computation protocol for NC^1 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 NC^1 computations.

4 3-Round Oblivious Transfer

In this section, we give a three-round oblivious transfer protocol that satisfies standard simulation security against malicious senders and super-polynomial simulation security against malicious receivers. We give the syntax of this protocol and the formal description of its security properties in Section 4.1. We then describe the building blocks used in the construction in Section 4.2. We give the construction in Section 4.3 and the proof of security in Section 4.4. Finally, in Section 4.5, we describe how to modify the construction such that the protocol satisfies certain additional properties needed for our applications.

4.1 Definition

Syntax. A three-round oblivious transfer protocol is given by a tuple of algorithms $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3, \mathsf{out}_{\mathsf{OT}})$. In the first round, the sender with input two strings (M_0, M_1) runs $\mathsf{OT}_1(1^\lambda, (M_0, M_1))$ to obtain the first round message msg_1 and sends this to the receiver. In the second round, the receiver with input $c \in \{0,1\}$ and uniform random tape $r \in \{0,1\}^*$ runs $\mathsf{OT}_2(\mathsf{msg}_1, c; r)$ to obtain msg_2 and sends this to the sender. In the final round, the sender runs $\mathsf{OT}_3(\mathsf{msg}_2, (M_0, M_1))$ to obtain msg_3 and forwards this to the receiver. Finally, the receiver runs $\mathsf{out}_{\mathsf{OT}}(\mathsf{msg}_1, \mathsf{msg}_3, (c, r))$ and obtains M_c.

In the following (and in the rest of the paper), we use $\mathsf{View}_A(\langle A(1^\lambda, x), B(1^\lambda, y)\rangle)$ (resp. View_B) to denote the view of $A(1^\lambda, x)$ (resp. $B(1^\lambda, y)$) in its interaction with $B(1^\lambda, y)$ (resp. $A(1^\lambda, x)$). Similarly, we use out_A (resp. out_B) to denote the output computed by $A(1^\lambda, x)$ (resp. $B(1^\lambda, y)$) at the end of the protocol.

Definition 4.1. *A three-round oblivious transfer protocol* $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3, \mathsf{out}_{\mathsf{OT}})$ *is said to satisfy standard security against malicious senders and super-polynomial simulation security against malicious receivers if:*

- ***Security against Malicious Senders.*** *There exists an expected PPT machine* Sim_S *such that for every non-uniform PPT adversary* $\mathcal{A}$ *corrupting the sender and for any choice of receiver's input* $c \in \{0,1\}$*, we have:*

$$\Big\{\big(\mathsf{View}_{\mathcal{A}}(\langle R(1^\lambda, c), \mathcal{A}(1^\lambda)\rangle), \mathsf{out}_R(\langle R(1^\lambda, c), \mathcal{A}(1^\lambda)\rangle)\big)\Big\} \approx_c \Big\{(\mathsf{View}_{\mathcal{A}}, M_c) : (\mathsf{View}_{\mathcal{A}}, (M_0, M_1)) \leftarrow (\mathsf{Sim}_S)^{\mathcal{A}}(1^\lambda)\Big\}$$

- ***Super-Polynomial Simulation Security against Malicious Receivers.*** *There exist an expected PPT machine* $\mathsf{Sim}_R = (\mathsf{Sim}_R^1, \mathsf{Sim}_R^2)$ *and a super-polynomial time machine* Ext_R *such that for every non-uniform PPT adversary* $\mathcal{A}$ *corrupting the receiver and for any choice of sender inputs* (M_0, M_1)*, we have:*

$$\Big\{\mathsf{View}_{\mathcal{A}}(\langle \mathcal{A}(1^\lambda), S(1^\lambda, (M_0, M_1))\rangle)\Big\} \approx_c \Big\{\mathsf{View}_{\mathcal{A}} : (\mathsf{msg}_1, \mathsf{st}) \leftarrow (\mathsf{Sim}_R^1)^{\mathcal{A}}(1^\lambda), \mathsf{msg}_2 \leftarrow \mathcal{A}(1^\lambda, \mathsf{msg}_1), c \leftarrow \mathsf{Ext}_R(\mathsf{msg}_2), \mathsf{View}_{\mathcal{A}} \leftarrow (\mathsf{Sim}_R^2)^{\mathcal{A}}(\mathsf{st}, M_c)\Big\}$$

The main theorem that we prove in this section is:

Theorem 4.2. *For some $\epsilon > 0$, assume black-box access to a two-round oblivious transfer protocol with super-polynomial time simulation security against malicious receivers and $(2^{\lambda^\epsilon}, 2^{-\lambda^\epsilon})$-indistinguishability-based security against malicious senders. Then, there exists a three-round OT protocol that satisfies Definition 4.1.*

4.2 Building Blocks

We now describe the building blocks used in our construction.

2-round, k-out-of-m Oblivious Transfer. A tuple of algorithms $(\mathsf{OT}_1^{(k,m)}, \mathsf{OT}_2^{(k,m)}, \mathsf{out}_{\mathsf{OT}^{(k,m)}})$ implementing the k-out-of-m oblivious transfer functionality that satisfies the following properties:

- **Correctness.** For every set K with elements from $[m]$ and cardinality k, any sequence of m strings $(s_1, \ldots, s_m)$ and for any choice of random tape r of receiver, we have:

$$\Pr[\mathsf{out}_{\mathsf{OT}^{(k,m)}}(\mathsf{otm}_1, \mathsf{otm}_2, (K, r)) = \{s_i\}_{i \in K}] = 1$$

  where $\mathsf{otm}_1 := \mathsf{OT}_1^{(k,m)}(1^\lambda, K; r)$ and $\mathsf{otm}_2 \leftarrow \mathsf{OT}_2^{(k,m)}(\mathsf{otm}_1, (s_1, \ldots, s_m))$.

- **Super-Polynomial Simulation Security against Malicious Receivers.** There exits a super-polynomial time algorithm Ext that runs in time $T_1(\lambda)$ (abbreviated as T_1) such that for any non-uniform PPT $\mathcal{A}$ and for any sender's input $(s_1, \ldots, s_m)$, we have:

$$\left\{\mathsf{OT}_2^{(k,m)}(\mathsf{otm}_1, (s_1, \ldots, s_m)) : \mathsf{otm}_1 \leftarrow \mathcal{A}(1^\lambda)\right\} \approx_c$$
$$\left\{\mathsf{OT}_2^{(k,m)}(\mathsf{otm}_1, (s_1^*, \ldots, s_m^*)) : \mathsf{otm}_1 \leftarrow \mathcal{A}(1^\lambda), K \leftarrow \mathsf{Ext}(\mathsf{otm}_1), \{s_i^* = s_i\}_{i \in K}, \{s_i^* = \perp\}_{i \notin K}\right\}$$

  where K output by Ext is of size at most k.

- **Indistinguishability against Malicious Senders.** For any PPT adversary $\mathcal{A}$ corrupting the sender S and every sets K_0, K_1 with elements from $[m]$ and cardinality k, we have:

$$\left\{\mathsf{View}_\mathcal{A}(\langle R(1^\lambda, K_0), \mathcal{A}(1^\lambda)\rangle)\right\} \approx_c \left\{\mathsf{View}_\mathcal{A}(\langle R(1^\lambda, K_1), \mathcal{A}(1^\lambda)\rangle)\right\}$$

As a special case, we use $\mathsf{OT}^{(1,2)}$ to denote an 1-out-of-2 OT protocol. $\mathsf{OT}^{(k,m)}$ can be constructed based on any 1-out-of-2 OT protocol that satisfies super-polynomial time simulation security against malicious receivers [AIR01, NP01, Kal05, HK12, BD18] via garbled circuits [Yao86].

Extractable Commitment. A three round extractable commitment scheme $(\mathsf{ECom}_1, \mathsf{ECom}_2, \mathsf{ECom}_3)$ that is (T_2, ϵ)-hiding against malicious receivers and satisfies over extraction (see Definition 3.5).

Pairwise Verifiable Secret Sharing. A pairwise verifiable, k-out-of-m secret sharing scheme $(\mathsf{Share}, \mathsf{Rec})$ that has the following properties:

- **Correctness.** For any secret s,

$$\Pr[\mathsf{Rec}(\mathsf{Sh}_1, \ldots, \mathsf{Sh}_m) = s : (\mathsf{Sh}_1, \ldots, \mathsf{Sh}_m) \leftarrow \mathsf{Share}(s)] = 1$$

- **Perfect Secrecy.** For any two secrets s_0, s_1 and for any subset $K \subset [m]$ of size k, we have:

$$\Big\{\mathsf{Sh}_K : (\mathsf{Sh}_1, \ldots, \mathsf{Sh}_m) \leftarrow \mathsf{Share}(s_0)\Big\} \equiv \Big\{\mathsf{Sh}_K : (\mathsf{Sh}_1, \ldots, \mathsf{Sh}_m) \leftarrow \mathsf{Share}(s_1)\Big\}$$

- **Pairwise Verifiability.** If there exists a set $K \subseteq [m]$ of size at least $S(m,k)$ such that for every $i, j \in K$, $(\mathsf{Sh}_i, \mathsf{Sh}_j)$ are pairwise consistent, then for any value of $\mathsf{Sh}_{[m]\setminus K}$, the output of $\mathsf{Rec}(\mathsf{Sh}_1, \ldots, \mathsf{Sh}_m)$ is the same.

Such pairwise verifiable secret sharing can be constructed from Bivariate Shamir secret sharing scheme.

Setting the Parameters. To instantiate the pairwise verifiable secret sharing, we set $m = 8\lambda$, $k = m/4$, and $S(m,k) = m - k$. To instantiate the extractable commitment, we set $T_2 \geq T_1 \cdot \mathsf{poly}(\binom{m}{k} \cdot \lambda)$ and ϵ is such that $\epsilon \cdot \binom{m}{k} \leq \mathsf{negl}(\lambda)$.

4.3 Construction

We give the formal description of the construction in Figure 2 and show in the next subsection that it satisfies Definition 4.1.

4.4 Proof of Security

In section 4.4.1, we show that the protocol given in Figure 2 satisfies security against malicious senders and in section 4.4.2, we show that this protocol satisfies super-polynomial simulation security against malicious receivers.

4.4.1 Security against Malicious Senders

We start with the description of the simulator Sim_S.

Description of Sim_S. Let $\mathcal{A}$ be the adversary that corrupts the sender. Sim_S does the following:

1. It initializes $\mathcal{A}$ with a uniform random tape.
2. It executes the protocol honestly using the receiver input set to some $c' \in \{0,1\}$ and rewinds the second and third rounds to extract $\{(\mathsf{Sh}^i_0, \mathsf{Sh}^i_1, s_i)\}_{i \in [m]}$ from the extractable commitment scheme.
3. It initializes an empty set I.
4. For each $i \in [m]$:
   (a) It checks if $\mathsf{otm}^i_2 := \mathsf{OT}^{(1,2)}_2(\mathsf{otm}_1, (\mathsf{Sh}^i_0, \mathsf{Sh}^i_1); s_i)$.
   (b) If the check fails, it adds i to the set I.
5. If $|I| > \lambda$, it outputs $\mathsf{View}_{\mathcal{A}}$ and sets (M_0, M_1) to $(\perp, \perp)$.
6. Else, it constructs an inconsistency graph $G = ([m], E)$ where $(i,j) \in E$ iff:

The input of the receiver is a bit $c \in \{0,1\}$ and the input of the sender are two λ-bit strings (M_0, M_1).

- **Round-1:** S does the following:
  1. For each $b \in \{0,1\}$, it samples a uniform string $X_b \leftarrow \{0,1\}^\lambda$ and computes $(\mathsf{Sh}_b^1, \ldots, \mathsf{Sh}_b^m) \leftarrow \mathsf{Share}(X_b)$.
  2. For each $i \in [m]$,
     (a) It chooses $r_i \leftarrow \{0,1\}^*$ as the randomness for the extractable commitment scheme.
     (b) It chooses $s_i \leftarrow \{0,1\}^*$ as the sender randomness for $\mathsf{OT}^{(1,2)}$.
     (c) It computes $\mathsf{Com}_1^i := \mathsf{ECom}_1(1^\lambda, (\mathsf{Sh}_0^i, \mathsf{Sh}_1^i, s_i); r_i)$.
  3. It sends $\{\mathsf{Com}_1^i\}_{i\in[m]}$ to R.
- **Round-2:** R does the following:
  1. It chooses a random set K which is a subset of $[m]$ of size k.
  2. It computes $\mathsf{otm}_1 \leftarrow \mathsf{OT}_1^{(1,2)}(1^\lambda, c)$.
  3. It computes $\overline{\mathsf{otm}}_1 \leftarrow \mathsf{OT}_1^{(k,m)}(1^\lambda, K)$.
  4. For each $i \in [m]$, it computes $\mathsf{Com}_2^i \leftarrow \mathsf{ECom}_2(\mathsf{Com}_1^i)$.
  5. It sends $\left(\mathsf{otm}_1, \overline{\mathsf{otm}}_1, \{\mathsf{Com}_2^i\}_{i\in[m]}\right)$ to S.
- **Round-3:** S does the following:
  1. For each $i \in [m]$,
     (a) It computes $\mathsf{otm}_2^i := \mathsf{OT}_2^{(1,2)}(\mathsf{otm}_1, (\mathsf{Sh}_0^i, \mathsf{Sh}_1^i); s_i)$.
     (b) It computes $\mathsf{Com}_3^i := \mathsf{ECom}_3(\mathsf{Com}_2^i, (\mathsf{Sh}_0^i, \mathsf{Sh}_1^i, s_i); r_i)$.
  2. It computes $\overline{\mathsf{otm}}_2 \leftarrow \mathsf{OT}_2^{(k,m)}(\overline{\mathsf{otm}}_1, (r_1, \ldots, r_m))$.
  3. It sets $\mathsf{ct}_0 := X_0 \oplus M_0$ and $\mathsf{ct}_1 := X_1 \oplus M_1$.
  4. It sends $\left(\overline{\mathsf{otm}}_2, (\mathsf{ct}_0, \mathsf{ct}_1), \{\mathsf{otm}_2^i, \mathsf{Com}_3^i\}_{i\in[m]}\right)$ to R.
- **Output Phase.** R does the following:
  1. For each $i \in [m]$, it runs $\mathsf{Verify}(\mathsf{Com}_1^i, \mathsf{Com}_2^i, \mathsf{Com}_3^i)$ and if any of the checks fail, it aborts.
  2. It recovers $\{r_i\}_{i\in K}$ from $\overline{\mathsf{otm}}_2$.
  3. For each $i \in K$,
     (a) It recovers $(\mathsf{Sh}_0^i, \mathsf{Sh}_1^i, s_i)$ from $(\mathsf{Com}_1^i, \mathsf{Com}_3^i)$ using the randomness r_i.
     (b) It checks if $\mathsf{otm}_2^i = \mathsf{OT}_2^{(1,2)}(\mathsf{otm}_1, (\mathsf{Sh}_0^i, \mathsf{Sh}_1^i); s_i)$. If this check fails, it aborts.
  4. For every $(i,j) \in K \times K$, it checks if:
     (a) $(\mathsf{Sh}_b^i, \mathsf{Sh}_b^j)$ are pairwise consistent for each $b \in \{0,1\}$.
     (b) If any of the checks fail, it aborts.
  5. For each $i \in [m]$, it recovers Sh_c^i from otm_2^i.
  6. It computes $X_c := \mathsf{Rec}(\mathsf{Sh}_c^1, \ldots, \mathsf{Sh}_c^m)$ and outputs $\mathsf{ct}_c \oplus X_c$.

Figure 2: Construction of Three-Round Oblivious Transfer

(a) If there exists $b \in \{0,1\}$ such that $(\mathsf{Sh}_b^i, \mathsf{Sh}_b^j)$ are pairwise inconsistent.

7. It computes a 2-approximation for the minimum vertex cover on the graph G and let I' denote

such a vertex cover.

8. If $|I'| > \lambda$, then it outputs $\mathsf{View}_{\mathcal{A}}$ and sets (M_0, M_1) to $(\perp, \perp)$.

9. It then performs the same checks that are run by the honest receiver and if any of the checks fail, it sets (M_0, M_1) to $(\perp, \perp)$.

10. Else, for each $b \in \{0, 1\}$, it computes $X_b := \mathsf{Rec}(\mathsf{Sh}_b^1, \ldots, \mathsf{Sh}_b^m)$. It then outputs the $\mathsf{View}_{\mathcal{A}}$ and sets (M_0, M_1) to $(\mathsf{ct}_0 \oplus X_0, \mathsf{ct}_1 \oplus X_1)$.

Proof of Indistinguishability.

- $\underline{\mathsf{Hyb}_0}$: This hybrid corresponds to $\big(\mathsf{View}_{\mathcal{A}}(\langle R(1^\lambda, b), \mathcal{A}(1^\lambda)\rangle), \mathsf{out}_R(\langle R(1^\lambda, b), \mathcal{A}(1^\lambda)\rangle)\big)$.

- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we run the extractor for the extractable commitment scheme and extract $\{(\mathsf{Sh}_0^i, \mathsf{Sh}_1^i, s_i)\}_{i \in [m]}$. Since there is no change in either the distribution of the messages in view of $\mathcal{A}$ or in the computation of the output by the honest receiver, it follows that $\mathsf{Hyb}_0 \equiv \mathsf{Hyb}_1$.

- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we compute the set I as described in the simulation and if $|I| > \lambda$, we output $\perp$ instead of M_b. We show that $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$ in Lemma 4.3 based on the indistinguishability against malicious senders property of $\mathsf{OT}^{(k,m)}$.

- $\underline{\mathsf{Hyb}_3}$: In this hybrid,

  1. We compute the inconsistency graph G as described in the simulator.
  2. We compute a 2-approximation of the minimum vertex cover and obtain I'.
  3. If $|I'| > \lambda$, then we output $\perp$.

  In Lemma 4.7, we show that $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$.

- $\underline{\mathsf{Hyb}_4}$: In this hybrid, we compute X_c as $\mathsf{Rec}(\mathsf{Sh}_c^1, \ldots, \mathsf{Sh}_c^m)$ where $(\mathsf{Sh}_c^1, \ldots, \mathsf{Sh}_c^m)$ are the shares extracted from the extractable commitment scheme. In Lemma 4.9, we show that Hyb_3 and Hyb_4 are identically distributed.

- $\underline{\mathsf{Hyb}_5}$: In this hybrid, we change otm_1 to be computed as $\mathsf{OT}_1^{(1,2)}(1^\lambda, c')$. It follows from the indistinguishability against malicious senders property of oblivious transfer that Hyb_4 is computationally indistinguishable from Hyb_5. Note that Hyb_5 is identically distributed to the $(\mathsf{View}_{\mathcal{A}}, M_c)$ where $(\mathsf{View}_{\mathcal{A}}, (M_0, M_1))$ are output by Sim_S.

Lemma 4.3. *Assuming the indistinguishability against malicious sender property of* $\mathsf{OT}^{(k,m)}$*, we have* $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$.

Proof. Assume for the sake of contradiction that Hyb_1 and Hyb_2 can be distinguished with non-negligible advantage. Note that the only difference between Hyb_1 and Hyb_2 is that in Hyb_1, if $|I| > \lambda$ and if all the checks performed by the honest receiver pass, then we output M_c whereas in Hyb_2, we output $\perp$. Hence, if Hyb_1 and Hyb_2 are distinguishable then with non-negligible advantage, the following event happens with non-negligible probability:

1. $|I| > \lambda$.

2. All the checks performed by honest receiver passes. This in particular implies that the set K chosen by the receiver is such that $|K \cap I| = 0$.

We now argue that this will contradict the indistinguishability against malicious sender property of $\mathsf{OT}^{(k,m)}$. Consider the following reduction.

1. We interact with the challenger for the indistinguishability against malicious receiver property and give 2 randomly chosen sets K_0, K_1 of $[m]$ of size k as the challenge inputs to the receiver.
2. We receive $\overline{\mathsf{otm}}_1$ from the challenger and use it to complete the execution with $\mathcal{A}$ and extract $\{(\mathsf{Sh}_0^i, \mathsf{Sh}_1^i, s_i)\}_{i\in[m]}$ as in Hyb_1.
3. We compute the set I as described in Hyb_2.
4. If $|I| > \lambda$ and there exists a $b' \in \{0,1\}$ such that $|I \cap K_{b'}| = 0$ and $|I \cap K_{1-b'}| \neq 0$ then we output b'. Otherwise, we output a random bit b'.

To complete the proof, we argue that $\Pr[b' = b]$ (where b is the challenge bit in the above game) is at least $1/2 + \mathsf{non-negl}$. We show this through a sequence of claims.

Claim 4.4. *If* $|I| > \lambda$, *then* $\Pr[|K_{1-b} \cap I| = 0] \leq 2^{-O(\lambda)}$.

Proof. Note that with respect to the view of the adversary, the set K_{1-b} is a random subset of $[m]$ and cardinality k. Hence,

$$\begin{aligned}
\Pr[|K_{1-b} \cap I| = 0] &= \frac{\binom{m-|I|}{k}}{\binom{m}{k}} \\
&< \frac{\binom{m-\lambda}{k}}{\binom{m}{k}} \quad \text{(since } |I| > \lambda) \\
&= \frac{(m-\lambda)!}{m!}\frac{(m-k)!}{(m-\lambda-k)!} \\
&< (1-\lambda/m)^k \\
&< 2^{-O(\lambda)}
\end{aligned}$$

□

Let E be the event that $|I| > \lambda$ and $|I \cap K_b| = 0$. We note that by assumption, $\Pr[E]$ is non-negligible.

Claim 4.5. $|\Pr[b = b' | \overline{E}] - 1/2| \leq 2^{-O(\lambda)}$.

Proof. If $\overline{E}$ happens then either,

1. $|I| \leq \lambda$.
2. $|I| > \lambda$ and $|I \cap K_b| \neq 0$.

In the first case, $\Pr[b' = b] = 1/2$. In the second case, if $|I \cap K_{1-b}| = 0$, then $\Pr[b' = b] = 0$. But from Claim 4.4, the probability that this happens is $2^{-O(\lambda)}$. On the other hand, if $|I \cap K_{1-b}| \neq 0$, then $\Pr[b' = b] = 1/2$. Thus, $|\Pr[b' = b|\overline{E}] - 1/2| \leq 2^{-O(\lambda)}$. □

Claim 4.6. $\Pr[b = b'|E] \geq 1 - 2^{-O(\lambda)}$.

Proof. Note that if $|I \cap K_{1-b}| \neq 0$, then $b = b'$. From Claim 4.4, the probability that this event happens is $1 - 2^{-O(\lambda)}$. □

We are now ready to lower bound $\Pr[b = b']$.

$$\begin{aligned}
\Pr[b = b'] &= Pr[E]\Pr[b' = b|E] + \Pr[\overline{E}]\Pr[b' = b|\overline{E}] \\
&= \Pr[b' = b|\overline{E}] + \Pr[E](\Pr[b' = b|E] - \Pr[b' = b|\overline{E}]) \\
&\geq (1/2 - 2^{-O(\lambda)}) + \Pr[E](\Pr[b' = b|E] - (1/2 + 2^{-O(\lambda)})) \\
&\geq (1/2 - 2^{-O(\lambda)}) + \Pr[E](1/2 - 2^{-O(\lambda)})) \\
&\geq 1/2 + \mathsf{non\text{-}negl}
\end{aligned}$$

where the first inequality follows from Claim 4.5 and the second inequality follows from Claim 4.6.

□

Lemma 4.7. *Assuming the indistinguishability against malicious sender property of* $\mathsf{OT}^{(k,m)}$*, we have* $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$.

Proof. The proof of this lemma is very similar to the proof of Lemma 4.3. To complete the proof, it is sufficient to show the following claim.

Claim 4.8. *If* $|I'| > \lambda$*, then* $\Pr[\forall(i,j) \in K_{1-b} \times K_{1-b}$ *and* $b \in \{0,1\}, (\mathsf{Sh}_b^i, \mathsf{Sh}_b^j)$ *are pairwise consistent*$] \leq 2^{-O(\lambda)}$.

Proof. Let F be the event that $\forall(i,j) \in K_{1-b} \times K_{1-b}$ and $b \in \{0,1\}, (\mathsf{Sh}_b^i, \mathsf{Sh}_b^j)$ are pairwise consistent. If $|I'| > \lambda$, then the size of the minimum vertex cover V is of size $> \lambda/2$. From a well-known connection between maximum matching and minimum vertex cover, it now follows that there exists a maximum matching in G with $\lambda/4$ edges. Thus, if there exists at least edge in the matching where both end points are in K_{1-b}, then the event F does not happen. Via an identical argument given in [IKOS07, Theorem 4.1], we can upper bound the probability that event F happens as $2^{-O(\lambda)}$. □

□

Lemma 4.9. $\mathsf{Hyb}_3 \equiv \mathsf{Hyb}_4$.

Proof. Note that $|I \cup I'| < 2\lambda = k$. Hence, by the pairwise verifiability property of the secret sharing scheme, the output of the reconstruction procedure in Hyb_3 and Hyb_4 are exactly the same. □

4.4.2 Super-Polynomial Simulation Security against Malicious Receivers

We now give the descriptions of $\mathsf{Sim}_R^1, \mathsf{Sim}_R^2$, and Ext_R.

Description of Sim_R^1. Let $\mathcal{A}$ be the adversary corrupting the receiver. Sim_R^1 does the following:

1. It initializes $\mathcal{A}$ with a uniformly chosen random tape.
2. It runs the first round of the protocol honestly using randomly sampled (X_0, X_1).
3. It sets st to be its random tape.
4. It outputs the first round message and st.

Description of Ext_R: Ext_R parses the message from $\mathcal{A}$ as $(\mathsf{otm}_1, \overline{\mathsf{otm}}_1, \{\mathsf{Com}_2^i\}_{i\in[m]})$ and runs $\mathsf{Ext}_{\mathsf{OT}^{(1,2)}}(\mathsf{otm}_1)$ to obtain c and outputs it.

Description of Sim_R^2. Sim_R^2 does the following:

1. It uses the random tape in st to obtain the transcript of the first two rounds of the protocol.
2. To generate the final round protocol message, it computes $(\overline{\mathsf{otm}}_2, \{\mathsf{otm}_2^i, \mathsf{Com}_3^i\}_{i\in[m]})$ as per the protocol description.
3. It computes $C_0 = X_0 \oplus M_c$ and $C_1 = X_1 \oplus M_c$ and sends the last round message to $\mathcal{A}$.
4. It finally outputs the view of $\mathcal{A}$.

Proof of Indistinguishability. We show the real world view of the adversary and the one that is generated by Sim_R are computationally indistinguishable via a hybrid argument.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to $\mathsf{View}_{\mathcal{A}}(\langle \mathcal{A}(1^\lambda), S(M_0, M_1)\rangle)$.
- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we change $\overline{\mathsf{otm}}_2$ to be computed as $\mathsf{OT}_2^{(k,m)}(\overline{\mathsf{otm}}, (r_1^*, \ldots, r_m^*))$ where $r_i^* = r_i$ if $i \in K$ where $K := \mathsf{Ext}_{\mathsf{OT}^{(k,m)}}(\overline{\mathsf{otm}}_1)$ and is otherwise, set to $\perp$.

  We show that $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_0$ in Lemma 4.10.
- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we make the following changes.
  1. We repeat the following for $\lambda \cdot \binom{m}{k}$ iterations:
     (a) We randomly choose $K' \subset [m]$ of cardinality k before sending the first round message.
     (b) If the extracted K at the end of the second round is not equal to the guessed values K', then we go to the next iteration.
  2. If we fail in each of the iterations then we output a special symbol fail and abort.

  In Lemma 4.11, we show that $\mathsf{Hyb}_2 \approx_s \mathsf{Hyb}_1$.
- $\underline{\mathsf{Hyb}_3}$: In this hybrid, in each of the $\lambda \cdot \binom{m}{k}$ iterations and for each $i \notin K'$, we change $\{\mathsf{Com}_1^i, \mathsf{Com}_3^i\}$ to be computed as extractable commitment to some dummy message instead of commitment to $(\mathsf{Sh}_0^i, \mathsf{Sh}_1^i, s_i)$. In Lemma 4.12, we show that $\mathsf{Hyb}_2 \approx_{\epsilon\lambda\cdot\binom{m}{k}\cdot m} \mathsf{Hyb}_3$.
- $\underline{\mathsf{Hyb}_4}$: In this hybrid, we change otm_2^i for each $i \notin K$, to be computed as $\mathsf{OT}_2^{(1,2)}(\mathsf{otm}_1, \overline{\mathsf{Sh}}_0^i, \overline{\mathsf{Sh}}_1^i)$ where $c := \mathsf{Ext}_{\mathsf{OT}^{(1,2)}}(\mathsf{otm}_1)$, $\overline{\mathsf{Sh}}_c^i = \mathsf{Sh}_c^i$, and $\overline{\mathsf{Sh}}_{1-c}^i = \perp$. We show in Lemma 4.13 that $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_3$.
- $\underline{\mathsf{Hyb}_5}$: In this hybrid, we change ct_{1-c} as $X_{1-c} \oplus M_c$ where $c := \mathsf{Ext}_{\mathsf{OT}^{(1,2)}}(\mathsf{otm}_1)$. We note that Hyb_5 is identically distributed to Hyb_4 from the perfect secrecy of the pairwise verifiable secret sharing scheme.
- $\underline{\mathsf{Hyb}_6} - \underline{\mathsf{Hyb}_9}$: In these hybrids, we reverse the changes made in Hyb_4 to Hyb_1. Via identical arguments given above, these changes are indistinguishable except with advantage $\epsilon\lambda \cdot \binom{m}{\lambda} \cdot m + \mathsf{negl}(\lambda)$. Note that Hyb_9 is distributed identically to the output generated in the ideal world with $(\mathsf{Sim}_R^1, \mathsf{Ext}_R, \mathsf{Sim}_R^2)$.

Lemma 4.10. *Assuming the super-polynomial simulation security against malicious receivers of* $\mathsf{OT}^{(k,m)}$*, we have* $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_0$.

Proof. Assume for the sake of contradiction that Hyb_1 and Hyb_0 are computationally distinguishable. We now give a reduction that breaks the super-polynomial simulation security against malicious receivers.

1. We start the interaction with the external challenger and provide $(r_1, \ldots, r_m)$ as the challenge sender inputs.

2. We generate the first round message of the protocol as in Hyb_2 and receive $(\mathsf{otm}_1, \overline{\mathsf{otm}}_1, \{\mathsf{Com}_2^i\}_{i\in[m]})$ from $\mathcal{A}$ in the second round.

3. We forward $\overline{\mathsf{otm}}_1$ to the external challenger and obtain $\overline{\mathsf{otm}}_2$ in return.

4. We generate the rest of the round-3 protocol messages as in Hyb_3 and complete the execution with $\mathcal{A}$.

Note that if the obtained $\overline{\mathsf{otm}}$ contains the sender messages $(r_1, \ldots, r_m)$, then the view of the adversary generated by the reduction is identical to Hyb_0. Otherwise, the view of $\mathcal{A}$ is identical to Hyb_1. Thus, if these two hybrids are distinguishable, we obtain a reduction that breaks the super-polynomial simulation security against malicious receivers for the oblivious transfer. □

Lemma 4.11. $\mathsf{Hyb}_2 \approx_s \mathsf{Hyb}_1$.

Proof. Note that the only difference between Hyb_1 and Hyb_2 is that sometimes, Hyb_2 may output the special symbol fail. To show that $\mathsf{Hyb}_1 \approx_s \mathsf{Hyb}_2$, it is sufficient to show that the probability that in Hyb_2, the symbol fail is output is negligible.

We first note that in each iteration, the probability that the guessed value is incorrect is $1 - \frac{1}{\binom{m}{k}}$. Thus, the probability we fail to guess correctly in each of the $\lambda\binom{m}{k}$ iterations is given by $(1 - \frac{1}{\binom{m}{k}})^{\lambda\binom{m}{k}} \leq 2^{-O(\lambda)}$. □

Lemma 4.12. *Assuming the* (T_2, ϵ)*-hiding property of the extractable commitment scheme, we have* $\mathsf{Hyb}_2 \approx_{\epsilon\cdot\lambda\cdot\binom{m}{k}\cdot m} \mathsf{Hyb}_3$.

Proof. We consider each iteration described in Hyb_2 and show that in each of these iterations, the changes described in Hyb_3 cannot be distinguished from Hyb_2 except with $m \cdot \epsilon$ advantage. We show this for the first iteration and the case for any general iteration is identical.

By a standard averaging argument, there exists two intermediate hybrids $\mathsf{Hyb}_{2,j}, \mathsf{Hyb}_{2,j-1}$ (described below) such that they are computationally distinguishable with advantage ϵ. We now give the description of the two hybrids. Let K' be the guessed value in this iteration. In both hybrids, for every $j' < j$ and if $j' \notin K'$, we generate $\{\mathsf{Com}_1^{j'}, \mathsf{Com}_3^{j'}\}$ as a commitment to some dummy message. In both hybrids, for every $j' > j$ or if $j' < j$ and $j' \in K'$, we generate $\{\mathsf{Com}_1^{j'}, \mathsf{Com}_3^{j'}\}$ as a commitment to $(\mathsf{Sh}_0^{j'}, \mathsf{Sh}_1^{j'}, s_{j'})$. The only difference between these two hybrids in the generation of the j-th commitment. Specifically, if $j \notin K'$, we generate $\{\mathsf{Com}_1^j, \mathsf{Com}_3^j\}$ as a commitment to some dummy message in $\mathsf{Hyb}_{2,j}$ and in $\mathsf{Hyb}_{2,j-1}$, we generate it as a commitment to $(\mathsf{Sh}_0^j, \mathsf{Sh}_1^j, s_j)$. We now use a distinguisher against $\mathsf{Hyb}_{2,j}$ and $\mathsf{Hyb}_{2,j-1}$ to contradict the computational hiding property of the extractable commitment scheme. Note that if $j \in K'$, then $\mathsf{Hyb}_{2,j}$ and $\mathsf{Hyb}_{2,j-1}$ are identically distributed. Hence, in the rest of the proof, we assume w.l.o.g. that $j \notin K'$.

We interact with the challenger for the hiding game of the extractable commitment scheme and give two messages $(\mathsf{Sh}_0^j, \mathsf{Sh}_1^j, s_j)$ and a dummy message. We obtain the first round extractable commitment message Com_1^j and generate the rest of the protocol messages as in $\mathsf{Hyb}_{2,j-1}$ and send it to $\mathcal{A}$. On obtaining the second round message from $\mathcal{A}$, we forward Com_2^j to the challenger and obtain the third round message Com_3^j (if the extracted value of K is same as the guessed value). We use this to generate the final round message in the protocol as in $\mathsf{Hyb}_{2,j-1}$.

Note that if $\{\mathsf{Com}_1^j, \mathsf{Com}_3^j\}$ are commitments to $(\mathsf{Sh}_0^j, \mathsf{Sh}_1^j, s_j)$, then the view of the adversary is identically distributed to $\mathsf{Hyb}_{3,j-1}$. Otherwise, it is distributed identically to $\mathsf{Hyb}_{3,j}$. Furthermore, the running time of the above reduction is upper bounded by $T_1 \cdot \mathsf{poly}(\lambda, \binom{m}{k})$. Thus, the reduction breaks (T_2, ϵ)-hiding property of the extractable commitment scheme. □

Lemma 4.13. *Assuming the super-polynomial simulation security against malicious receivers of* $\mathsf{OT}^{(1,2)}$*, we have* $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$.

Proof. Assume for the sake of contradiction that Hyb_3 and Hyb_4 are computationally distinguishable. We now give a reduction that breaks the super-polynomial simulation security against malicious receivers.

1. We non-uniformly interact with the $\mathcal{A}$ as in Hyb_3 for the first two rounds until we reach an iteration where the guessed value K' is same as the extracted value.

2. We start the interaction with the external challenger and provide $\{(\mathsf{Sh}_0^i, \mathsf{Sh}_1^i)\}_{i \notin K}$ as the challenge sender inputs.

3. We forward otm_1 to the external challenger and obtain $\{\mathsf{otm}_2^i\}_{i \notin K}$ in return.

4. We generate the rest of the round-3 protocol messages as in Hyb_3 and complete the execution with $\mathcal{A}$.

Note that if the obtained $\{\overline{\mathsf{otm}}_2^i\}_{i \notin K}$ contain the sender inputs $(\mathsf{Sh}_0^i, \mathsf{Sh}_1^i)$, then the view of the adversary generated by the reduction is identical to Hyb_3. Otherwise, the view of $\mathcal{A}$ is identical to Hyb_4. Thus, we obtain a reduction that breaks the super-polynomial simulation security against malicious receivers for the oblivious transfer. □

4.5 Additional Properties

For our application to constructing round-optimal MPC protocols, we need this 3-round OT to satisfy certain additional properties. We mention these properties below and explain how to achieve these properties by suitably modifying the protocol.

k-Rewinding Sender Security. For our applications, we need security against malicious receivers that may rewind the second and third rounds of the honest sender k times for some constant k. We call an oblivious transfer protocol that is secure against such a stronger receiver as satisfying k-rewinding sender security. We now describe how to modify the construction given in Figure 2 to satisfy this stronger property in the simultaneous message exchange setting.[9] In the first round, in addition to sender S sending $\{\mathsf{Com}_1^i\}_{i\in[m]}$, the receiver R in parallel computes $(\mathsf{otm}_1, \overline{\mathsf{otm}}_1)$ as described in the protocol and sends them to the sender. In the second round, the receiver only computes $\{\mathsf{Com}_2^i\}_{i\in[m]}$ and sends it to S. The third round and the output computation steps remain the same. It is now easy to see that if the extractable commitment ECom used by the sender additionally satisfies k-rewinding, (T_2, ϵ)-hiding against malicious receivers then the modified protocol satisfies k-rewinding sender security against malicious receivers with super-polynomial time simulation. Note that with the above modification, the second round message from the receiver is public coin.

Public-Coin Second Round Message from Receiver. Note that with the above described changes the second round message from the receiver is just public coins.

Two Properties of Sim_S.

1. We note that the view of the adversary generated by Sim_S is identically distributed to its view when interacting with an honest receiver with choice bit c'. Specifically, we can augment Sim_S to additionally take a bit c' as input and produce the view of adversary using the receiver's choice bit as c'. It directly follows from the hiding property $\mathsf{OT}^{(1,2)}$ that the output of $\mathsf{Sim}_S(1^\lambda, 0)$ is computationally indistinguishable to $\mathsf{Sim}_S(1^\lambda, 1)$.

2. We note that to extract the sender input (M_0, M_1), Sim_S rewinds and extracts from the extractable commitment. Thus, for a given first round message from the malicious sender and valid third round responses to two random second round challenges, Sim_S can use the above information to extract (M_0, M_1) except with negligible probability. If the extractable commitment on the other hand, requires t accepting transcripts (with valid third round messages), then Sim_S also needs t accepting transcripts with valid third round messages.

Existence of Straight-Line SPS extractor against Malicious Senders. We note that Sim_S described earlier rewinds the extractable commitment to extract the sender inputs in the OT protocol. If the extractable commitment is straight-line extractable in super-polynomial time (such a construction of extractable commitment is given in Appendix A), we get a super-polynomial time, straight-line simulator against malicious senders.

5 3-Round Secure Two-Party Computation

In this section, we give a construction of a three-round two-party computation protocol that satisfies standard simulation security against malicious senders and super-polynomial simulation security against malicious receivers for computing NC^1 circuits. In comparison to the previous section, we now construct a protocol that computes general two-party functionalities rather than just oblivious transfer. In Section 5.1, we give the syntax and formally specify security properties that the protocol needs to satisfy. In Section 5.3, we give the construction of this protocol and in Section 5.4, we

[9] In the simultaneous message exchange setting, both parties can send a message in each round. By default, we consider rushing adversaries and hence, in each round, the adversary sends its message only after it receives the particular round message from all the honest parties.

give the proof of security. In Section 5.5, we show how to add certain additional properties such as k-rewinding sender security.

5.1 Syntax and Definition

Syntax. A three-round protocol Π for computing a function f is given by a tuple of algorithms $(\Pi_1, \Pi_2, \Pi_3, \mathsf{out}_\Pi)$. In the first round, the sender with input y runs $\Pi_1(1^\lambda, y)$ to obtain the first round message msg_1 and sends this to the receiver. In the second round, the receiver with input x and uniform random tape $r \in \{0,1\}^*$ runs $\Pi_2(\mathsf{msg}_1, x; r)$ to obtain msg_2 and sends this to the sender. In the final round, the sender runs $\Pi_3(\mathsf{msg}_2, y)$ to obtain msg_3 and forwards this to the receiver. Finally, the receiver runs $\mathsf{out}_\Pi(\mathsf{msg}_1, \mathsf{msg}_3, (x, r))$ and obtains $f(x, y)$.

Definition 5.1. *A two-party protocol $\Pi = (\Pi_1, \Pi_2, \Pi_3, \mathsf{out}_\Pi)$ for computing a function $f \in \mathsf{NC}^1$ is said to satisfy standard simulation security against malicious senders and super-polynomial simulation security against malicious receivers if the following three properties hold:*

- ***Delayed Function Selection.*** *We require that Π_1 and Π_2 to only depend on the size of the function f that is to be securely computed and is otherwise, independent of its description. Π_3 takes the description of the function f explicitly and generates the final round message in the protocol.*
- ***Security against Malicious Senders:*** *For every PPT adversary $\mathcal{A}$ corrupting the sender there exists an expected PPT machine Sim_S such that for every receiver input $x \in \{0,1\}^n$, we have:*

$$\Big\{\big(\mathsf{View}_\mathcal{A}(\langle R(1^\lambda, x), \mathcal{A}(1^\lambda)\rangle), \mathsf{out}_R(\langle R(1^\lambda, x), \mathcal{A}(1^\lambda)\rangle)\big)\Big\} \approx_c \Big\{(\mathsf{View}_\mathcal{A}, f(x, y)) : (\mathsf{View}_\mathcal{A}, y) \leftarrow (\mathsf{Sim}_S)^\mathcal{A}(1^\lambda)\Big\}$$

- ***Super-Polynomial Simulation Security against Malicious Receivers:*** *For every PPT adversary $\mathcal{A}$ corrupting the receiver, there exists a super-polynomial time machine Ext and an expected PPT machine $\mathsf{Sim}_R = (\mathsf{Sim}_R^1, \mathsf{Sim}_R^2)$ such that for every sender's input $y \in \{0,1\}^n$, we have:*

$$\Big\{\mathsf{View}_\mathcal{A}(\langle \mathcal{A}(1^\lambda), S(1^\lambda, y)\rangle)\Big\} \approx_c \Big\{\mathsf{View}_\mathcal{A} : (\mathsf{msg}_1, \mathsf{st}) \leftarrow (\mathsf{Sim}_R^1)^\mathcal{A}(1^\lambda), \mathsf{msg}_2 \leftarrow \mathcal{A}(1^\lambda, \pi_1), (x, \mathsf{st}') \leftarrow \mathsf{Ext}(\pi_2, \mathsf{st}), \mathsf{View}_\mathcal{A} \leftarrow (\mathsf{Sim}_R^2)^\mathcal{A}(\mathsf{st}, \mathsf{st}', f(x, y))\Big\}$$

5.2 Building Blocks

The construction makes use of the following building blocks:

- **Outer MPC Protocol.** A two-round two clients, m-server MPC protocol $(\Phi_1, \Phi_2, \mathsf{out}_\Phi)$ for computing NC^1 functionalities that satisfies μ-statistical security with selective abort against adversary corrupting upto t servers and one of the clients. We set $t = \lambda$ and $m = 3t + 1$. We require this protocol to satisfy perfect first round message indistinguishability, meaning that for any input of the honest client, the distribution of the first round message sent to the corrupted servers from the honest client is identically distributed to the first round message generated by the simulator Sim_Φ on behalf of this client. The construction given in [IKP10, Pas12] satisfies this property. We also need this protocol to satisfy the property that Φ_1 only depends on the size of the functionality f to be securely computed and is otherwise, independent of its description. This property can be generically added by considering a dummy client that holds the description of the function and the messages sent by this client are generated using some default randomness.

- **2-round, k-out-of-m Oblivious Transfer.** $(\mathsf{OT}_1^{(k,m)}, \mathsf{OT}_2^{(k,m)}, \mathsf{out}_{\mathsf{OT}^{(k,m)}})$ implementing the k-out-of-m oblivious transfer functionality from Section 4.2. Let $T_1(\lambda)$ be an upper bound on the running time of $\mathsf{Ext}_{\mathsf{OT}^{(k,m)}}$.
- **Extractable Commitment.** A three round extractable commitment scheme $(\mathsf{ECom}_1, \mathsf{ECom}_2, \mathsf{ECom}_3)$ that is (T_2, ϵ)-hiding against malicious receivers and satisfies over extraction (see Definition 3.5).
- **Garbled Circuits.** A circuit garbling scheme $(\mathsf{Garble}, \mathsf{Eval})$.
- **Three-round OT protocol.** $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3, \mathsf{out}_{\mathsf{OT}})$ satisfying Definition 4.1 that is $\delta(\lambda)$-secure against malicious receivers running in time $T_2(\lambda)$. Let $T_1(\lambda)$ be the upper bound on the running time of $\mathsf{Ext}_{\mathsf{OT}}$.

Setting the Parameters. We set $T_2 \geq T_1 \cdot \mathsf{poly}(\lambda \cdot \binom{m}{\lambda})$ and $\epsilon = \mu = \delta$ such that $\epsilon \cdot \binom{m}{k} \leq \mathsf{negl}(\lambda)$.

5.3 Construction

We give the description of the construction in Figure 3.

5.4 Proof of Security

In this subsection, we show that the above construction satisfies Definition 5.1.

5.4.1 Security against Malicious Senders

Description of Sim_S.

1. Sim_S defines an adversary $\mathcal{A}$ that does the following:
   (a) It interacts with $\mathcal{A}$ and obtains the first round message $\{\mathsf{Com}_1^i\}_{i\in[m]}, \{\mathsf{otm}_1^{i,j}\}_{i\in[m],j\in[k]}$. It forwards $\{\mathsf{otm}_1^{i,j}\}_{i\in[m],j\in[k]}$ externally.
   (b) It obtains $\{\mathsf{otm}_2^{i,j}\}_{i\in[m],j\in[k]}$ externally and computes $\mathsf{otm}_1, \{\mathsf{Com}_2^i\}_{i\in[m]}$ as described in the protocol. It runs $\mathcal{A}$ on $\{\mathsf{otm}_2^{i,j}\}_{i\in[m],j\in[k]}, \{\mathsf{Com}_2^i\}_{i\in[m]}, \mathsf{otm}_1$.
   (c) It obtains $\{\mathsf{Com}_3^i, \widetilde{\Phi}_{2,i}\}_{i\in[m]}, \{\mathsf{otm}_3^{i,j}, \overline{\mathsf{lab}}_{y_{i,j}}^{i,j}\}_{i\in[m],j\in[k]}, \mathsf{otm}_2$ from $\mathcal{A}$ and forwards $\{\mathsf{otm}_3^{i,j}\}_{i\in[m],j\in[k]}$ externally.
2. Sim_S runs $\mathsf{Sim}_S^{\mathsf{OT}}$ to generate the view of the adversary $\mathcal{A}'$ in the first three rounds along with the inputs $\{\mathsf{lab}_0^{i,j}, \mathsf{lab}_1^{i,j}\}_{i\in[m],j\in[k]}$. We extract the view of $\mathcal{A}$ from $\mathsf{View}_{\mathcal{A}'}$.

- **Round-1:** In the first round, S does the following:
  1. It computes $(y_1, \ldots, y_m) \leftarrow \Phi_1(y)$ where each $y_i \in \{0,1\}^k$.
  2. For each $i \in [m]$,
     (a) It samples $\{\mathsf{lab}_0^{i,j}, \mathsf{lab}_1^{i,j}\}$ and $\{\overline{\mathsf{lab}}_0^{i,j}, \overline{\mathsf{lab}}_1^{i,j}\}$ for each $j \in [k]$ uniformly from $\{0,1\}^\lambda \times \{0,1\}^\lambda$.
     (b) For each $j \in [k]$, it samples $s_{i,j} \leftarrow \{0,1\}^*$ as the randomness for an OT execution. It sets $s_i := (s_{i,1}, \ldots, s_{i,k})$.
     (c) It samples $r_i \leftarrow \{0,1\}^*$ as the randomness for an extractable commitment scheme.
     (d) It samples $t_i \leftarrow \{0,1\}^*$ as the randomness for generating a garbled circuit.
     (e) It computes $\mathsf{Com}_1^i := \mathsf{ECom}_1(1^\lambda, (y_i, s_i, t_i, \{\mathsf{lab}_b^{i,j}, \overline{\mathsf{lab}}_b^{i,j}\}_{j\in[k],b\in\{0,1\}}); r_i)$.
  3. For each $i \in [m]$ and $j \in [k]$,
     (a) It computes $\mathsf{otm}_1^{i,j} \leftarrow \mathsf{OT}_1(1^\lambda, (\mathsf{lab}_0^{i,j}, \mathsf{lab}_1^{i,j}); s_{i,j})$
  4. It sends $\{\mathsf{Com}_1^i\}_{i\in[m]}, \{\mathsf{otm}_1^{i,j}\}_{i\in[m],j\in[k]}$ to R.
- **Round-2:** In the second round, R does the following:
  1. It computes $(x_1, \ldots, x_m) \leftarrow \Phi_1(x)$ where each $x_i \in \{0,1\}^k$.
  2. It samples a random subset $K \subset [m]$ of size λ and computes $\mathsf{otm}_1 \leftarrow \mathsf{OT}_1^{(\lambda,m)}(1^\lambda, K)$.
  3. For each $i \in [m]$ and $j \in [k]$,
     (a) It computes $\mathsf{otm}_2^{i,j} \leftarrow \mathsf{OT}_2(\mathsf{otm}_1^{i,j}, x_{i,j})$.
  4. For each $i \in [m]$,
     (a) It computes $\mathsf{Com}_2^i \leftarrow \mathsf{ECom}_2(\mathsf{Com}_1^i)$.
  5. It sends $\{\mathsf{otm}_2^{i,j}\}_{i\in[m],j\in[k]}, \{\mathsf{Com}_2^i\}_{i\in[m]}, \mathsf{otm}_1$ to S.
- **Round-3:** In the final round, S does the following:
  1. For each $i \in [m]$,
     (a) It computes $\mathsf{Com}_3^i \leftarrow \mathsf{ECom}_3(\mathsf{Com}_2^i, (y_i, s_i, t_i, \{\mathsf{lab}_b^{i,j}, \overline{\mathsf{lab}}_b^{i,j}\}_{j\in[k],b\in\{0,1\}}); r_i)$.
     (b) It computes $\widetilde{\Phi}_{2,i} := \mathsf{Garble}(\Phi_{2,i}, \{\mathsf{lab}_b^{i,j}, \overline{\mathsf{lab}}_b^{i,j}\}_{j\in[k],b\in\{0,1\}}; t_i)$.
  2. For each $i \in [m]$ and $j \in [k]$,
     (a) It computes $\mathsf{otm}_3^{i,j} := \mathsf{OT}_3(\mathsf{otm}_2^{i,j}, (\mathsf{lab}_0^{i,j}, \mathsf{lab}_1^{i,j}); s_{i,j})$.
  3. It computes $\mathsf{otm}_2 \leftarrow \mathsf{OT}_2^{(\lambda,m)}(\mathsf{otm}_1, (r_1, \ldots, r_m))$.
  4. It sends $\{\mathsf{Com}_3^i, \widetilde{\Phi}_{2,i}\}_{i\in[m]}, \{\mathsf{otm}_3^{i,j}, \overline{\mathsf{lab}}_{y_{i,j}}^{i,j}\}_{i\in[m],j\in[k]}, \mathsf{otm}_2$ to R.
- **Output Computation:** To compute the output, R does the following:
  1. It recovers $\{r_i\}_{i\in K}$ from otm_2.
  2. For each $i \in K$,
     (a) It recovers $(y_i, s_i, t_i, \{\mathsf{lab}_b^{i,j}, \overline{\mathsf{lab}}_b^{i,j}\}_{j\in[k],b\in\{0,1\}})$ from $\{\mathsf{Com}_1^i, \mathsf{Com}_3^i\}$ using randomness r_i.
     (b) For each $j \in [k]$, it checks if $\mathsf{otm}_1^{i,j} := \mathsf{OT}_1(1^\lambda, (\mathsf{lab}_0^{i,j}, \mathsf{lab}_1^{i,j}); s_{i,j})$ and $\mathsf{otm}_3^{i,j} := \mathsf{OT}_2(\mathsf{otm}_2^{i,j}, (\mathsf{lab}_0^{i,j}, \mathsf{lab}_1^{i,j}); s_{i,j})$.
     (c) It checks if $\widetilde{\Phi}_{2,i} := \mathsf{Garble}(\Phi_{2,i}, \{\mathsf{lab}_b^{i,j}, \overline{\mathsf{lab}}_b^{i,j}\}_{j\in[k],b\in\{0,1\}}; t_i)$.
     (d) It checks if the received label is consistent with the extracted value $\overline{\mathsf{lab}}_{y_{i,j}}^{i,j}$ from $\{\mathsf{Com}_1^i, \mathsf{Com}_3^i\}$.
     (e) If any of the checks fail, it outputs $\perp$.
  3. Else, for each $i \in [m]$ and $j \in [k]$, it recovers $\mathsf{lab}_{x_{i,j}}^{i,j}$ from $\{\mathsf{otm}_1^{i,j}, \mathsf{otm}_3^{i,j}\}$.
  4. For each $i \in [m]$, it computes $\phi_i \leftarrow \mathsf{Eval}(\widetilde{\Phi}_{2,i}, \{\mathsf{lab}_{x_{i,j}}^{i,j}, \overline{\mathsf{lab}}_{y_{i,j}}^{i,j}\}_{i\in[m],j\in[k]})$.
  5. It outputs $\mathsf{out}_\Phi(\phi_1, \ldots, \phi_m)$.

Figure 3: Construction of the 3-Round Two-Party Computation Protocol

3. While running $\mathsf{Sim}_S^{\mathsf{OT}}$ on $\mathcal{A}'$, Sim_S additionally runs in parallel the extractor for the extractable commitment to obtain $\{(y_i, s_i, t_i, \{\mathsf{lab}_b^{i,j}, \overline{\mathsf{lab}}_b^{i,j}\}_{j\in[k], b\in\{0,1\}})\}_{i\in[m]}$.

4. Sim_S initializes an empty set C.

5. For each $i \in [m]$,

   (a) For each $j \in [k]$, it checks if $\mathsf{otm}_1^{i,j} := \mathsf{OT}_1(1^\lambda, (\mathsf{lab}_0^{i,j}, \mathsf{lab}_1^{i,j}); s_{i,j})$ and $\mathsf{otm}_3^{i,j} := \mathsf{OT}_2(\mathsf{otm}_2^{i,j}, (\mathsf{lab}_0^{i,j}, \mathsf{lab}_1^{i,j}); s_{i,j})$.

   (b) It checks if $\widetilde{\Phi}_{2,i} := \mathsf{Garble}(\Phi_{2,i}, \{\mathsf{lab}_b^{i,j}, \overline{\mathsf{lab}}_b^{i,j}\}_{j\in[k], b\in\{0,1\}}; t_i)$.

   (c) It checks if the received label is consistent with the value $\overline{\mathsf{lab}}_{y_{i,j}}^{i,j}$ extracted from $\{\mathsf{Com}_1^i, \mathsf{Com}_3^i\}$.

   (d) If any of the checks fail, it adds $\{i\}$ to C.

6. If $|C| > \lambda$, it aborts and instructs the ideal functionality to output $\perp$ to the receiver.

7. It then initializes the simulator Sim_Φ for the protocol Φ by corrupting the sender client and the set of servers indexed by the set C. For each $i \in C$, it obtains $\{x_i\}_{i\in C}$ as the first round message from the honest receiver client to the corrupted servers.

8. It sends $\{y_i\}_{i\notin C}$ to Sim_Φ as the first round message from the corrupted sender client to the honest servers. Sim_Φ queries the ideal functionality on input y and it forwards this to its own ideal functionality.

9. It performs the same checks that are run by the honest receiver. If any of the checks fail, we output $\perp$. If all the checks pass, for each $i \in C$, it obtains ϕ_i by running $\mathsf{Eval}(\widetilde{\Phi}_{2,i}, \{\mathsf{lab}_{x_{i,j}}^{i,j}, \overline{\mathsf{lab}}_{y_{i,j}}^{i,j}\}_{i\in[m], j\in[k]})$.

10. It sends $\{\phi_i\}_{i\in C}$ to Sim_Φ as the final round message from the corrupt servers to the honest receiver client. If Sim_Φ instructs the honest receiver to output $\perp$, it forwards this instruction to its trusted functionality. Otherwise, it instructs the trusted functionality to deliver output to the honest receiver.

Proof of Indistinguishability.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to $\big(\mathsf{View}_{\mathcal{A}}(\langle R(1^\lambda, x), \mathcal{A}(1^\lambda)\rangle), \mathsf{out}_R(\langle R(1^\lambda, x), \mathcal{A}(1^\lambda)\rangle)\big)$.

- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we define an adversary $\mathcal{A}'$ that corrupts the sender for the OT executions and works as follows:

  1. It interacts with $\mathcal{A}$ and obtains the first round message $\{\mathsf{Com}_1^i\}_{i\in[m]}, \{\mathsf{otm}_1^{i,j}\}_{i\in[m], j\in[k]}$. It forwards $\{\mathsf{otm}_1^{i,j}\}_{i\in[m], j\in[k]}$ externally.

  2. It obtains $\{\mathsf{otm}_2^{i,j}\}_{i\in[m], j\in[k]}$ externally and computes $\mathsf{otm}_1, \{\mathsf{Com}_2^i\}_{i\in[m]}$ as described in the protocol. It runs $\mathcal{A}$ on $\{\mathsf{otm}_2^{i,j}\}_{i\in[m], j\in[k]}, \{\mathsf{Com}_2^i\}_{i\in[m]}, \mathsf{otm}_1$.

  3. It obtains $\{\mathsf{Com}_3^i, \widetilde{\Phi}_{2,i}\}_{i\in[m]}, \{\mathsf{otm}_3^{i,j}, \overline{\mathsf{lab}}_{y_{i,j}}^{i,j}\}_{i\in[m], j\in[k]}, \mathsf{otm}_2$ from $\mathcal{A}$ and forwards $\{\mathsf{otm}_3^{i,j}\}_{i\in[m], j\in[k]}$ externally.

We run $\mathsf{Sim}_S^{\mathsf{OT}}$ on $\mathcal{A}'$ to obtain the $\mathsf{View}_{\mathcal{A}'}$ and $\{\mathsf{lab}_0^{i,j}, \mathsf{lab}_1^{i,j}\}_{i\in[m],j\in[k]}$. We extract the $\mathsf{View}_{\mathcal{A}}$ from $\mathsf{View}_{\mathcal{A}'}$ and compute the output of R using the internal randomness of $\mathcal{A}'$ and the extracted $\{\mathsf{lab}_0^{i,j}, \mathsf{lab}_1^{i,j}\}_{i\in[m],j\in[k]}$.We show in Lemma 5.2 that Hyb_0 and Hyb_1 are computationally indistinguishable.

- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we run the extractor for the extractable commitment scheme to obtain $\{(y_i, s_i, t_i, \{\mathsf{lab}_b^{i,j}, \overline{\mathsf{lab}}_b^{i,j}\}_{j\in[k],b\in\{0,1\}})\}_{i\in[m]}$. Since neither the view of the adversary nor the output of the honest receiver changes between Hyb_1 and Hyb_2, it follows that Hyb_1 and Hyb_2 are identically distributed.

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

  1. We initialize an empty set C.
  2. For each $i \in [m]$,
     (a) For each $j \in [k]$, we check if $\mathsf{otm}_1^{i,j} := \mathsf{OT}_1(1^\lambda, (\mathsf{lab}_0^{i,j}, \mathsf{lab}_1^{i,j}); s_{i,j})$ and $\mathsf{otm}_3^{i,j} := \mathsf{OT}_2(\mathsf{otm}_2^{i,j}, (\mathsf{lab}_0^{i,j}, \mathsf{lab}_1^{i,j}); s_{i,j})$.
     (b) We check if $\widetilde{\Phi}_{2,i} := \mathsf{Garble}(\Phi_{2,i}, \{\mathsf{lab}_b^{i,j}, \overline{\mathsf{lab}}_b^{i,j}\}_{j\in[k],b\in\{0,1\}}; t_i)$.
     (c) We check if the received label $\overline{\mathsf{lab}}_{y_{i,j}}^{i,j}$ is consistent with the values extracted from $\{\mathsf{Com}_1^i, \mathsf{Com}_3^i\}$.
     (d) If any of the checks fail, we add $\{i\}$ to C.
  3. If $|C| > \lambda$, we abort and output $\perp$ and otherwise, we continue as before.

  Via an identical argument as given in Lemma 4.3, we can show that Hyb_2 and Hyb_3 are computationally indistinguishable.

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

  1. We initialize the simulator Sim_Φ for the protocol Φ by corrupting the sender client and the set of servers indexed by the set C. For each $i \in C$, we obtain $\{x_i\}_{i\in C}$ as the first round message from the honest receiver client to the corrupted servers.
  2. We send $\{y_i\}_{i\notin C}$ to Sim_Φ as the first round message from the corrupted sender client to the honest servers. Sim_Φ queries the ideal functionality on input y and we record this value.
  3. We complete the execution with $\mathcal{A}$ and to compute the output, we first perform the same checks that are run by the honest receiver in the previous hybrid. If any of the checks fail, we output $\perp$. If all the checks pass, for each $i \in C$, we obtain ϕ_i by running $\mathsf{Eval}(\widetilde{\Phi}_{2,i}, \{\mathsf{lab}_{x_{i,j}}^{i,j}, \overline{\mathsf{lab}}_{y_{i,j}}^{i,j}\}_{i\in[m],j\in[k]})$.
  4. We send $\{\phi_i\}_{i\in C}$ to Sim_Φ as the final round message from the corrupt servers to the honest receiver client. If Sim_Φ instructs the honest receiver to output $\perp$, we output $\perp$ and otherwise, we instruct the honest receiver to output $f(x, y)$.

  In Lemma 5.3, we show that Hyb_3 and Hyb_4 are statistically indistinguishable. Note that Hyb_4 is identically distributed to output of ideal execution generated by Sim_S.

Lemma 5.2. *Assuming the security against malicious senders for the 3-round* OT *protocol, we have that* $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_0$.

Proof. Assume for the sake of contradiction that Hyb_1 and Hyb_0 are computationally distinguishable. We now give a reduction to the security against malicious senders for the 3-round OT protocol.

We interact with the external challenger and give $\{x_{i,j}\}_{i\in[m],j\in[k]}$ as the receiver choice bits. We construct the adversary $\mathcal{A}'$ as described in Hyb_1. We start the interaction with the external challenger by forwarding the messages from the external challenger to $\mathcal{A}'$ and forwarding the messages from $\mathcal{A}'$ to the challenger. At the end of the interaction, we obtain $\mathsf{View}_{\mathcal{A}'}$ along with $\{\mathsf{lab}^{i,j}_{x_{i,j}}\}_{i\in[m],j\in[k]}$. We compute the $\mathsf{View}_{\mathcal{A}}$ and the output of the honest receiver as described in Hyb_1. We output these two values.

Note that if $\mathsf{View}_{\mathcal{A}'}$ and $\{\mathsf{lab}^{i,j}_{x_{i,j}}\}_{i\in[m],j\in[k]}$ are generated using the real protocol execution, then the output of the reduction is identically distributed to Hyb_0. Otherwise, it is identically distributed to Hyb_1. Thus, if Hyb_0 and Hyb_1 are computationally distinguishable, then the above reduction breaks the security against malicious senders for the parallel version of the OT protocol and this is a contradiction. □

Lemma 5.3. *Assuming the statistical security of the protocol Φ, we have $\mathsf{Hyb}_3 \approx_s \mathsf{Hyb}_4$.*

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

We interact with the external challenger by giving x as the honest receiver input. We corrupt the sender client and the set of servers corresponding to the set C. We obtain $\{x_i\}_{i\in C}$ as the first round message from the receiver to the corrupt servers. We provide $\{y_i\}_{i\notin C}$ as the first round message from the corrupt sender to the honest servers. For each $i \in C$, we compute the second round message ϕ_i from the corrupt servers as described in Hyb_4. We send $\{\phi_i\}_{i\in C}$ to the external challenger. The external challenger replies with either $f(x,y)$ or abort and we instruct the honest receiver to output the same. The reduction finally outputs the view of the adversary $\mathcal{A}$ along with the output of the honest receiver.

Note that if the messages of the protocol Φ generated by the external challenger correspond to the real protocol messages, then the output of the reduction is identical to Hyb_3. Else, it is identically distributed to Hyb_4. It now follows that since Hyb_3 and Hyb_4 are statistically distinguishable, the above reduction breaks the security of the protocol Φ and this is a contradiction. □

5.4.2 Super-Polynomial Simulation Security against Malicious Receivers.

Description of Sim^1_R. Sim^1_R does the following:

1. It generates $(y_1,\ldots,y_m) \leftarrow \Phi_1(\mathbf{0})$ (where $\mathbf{0}$ is some default input).
2. For each $i \in [m]$,
   (a) It samples $\{\mathsf{lab}^{i,j}_0, \mathsf{lab}^{i,j}_1\}$ and $\{\overline{\mathsf{lab}}^{i,j}_0, \overline{\mathsf{lab}}^{i,j}_1\}$ for each $j \in [k]$ uniformly from $\{0,1\}^\lambda \times \{0,1\}^\lambda$.
   (b) For each $j \in [k]$, it samples $s_{i,j} \leftarrow \{0,1\}^*$ as the randomness for an OT execution. It sets $s_i := (s_{i,1},\ldots,s_{i,k})$.
   (c) It samples $r_i \leftarrow \{0,1\}^*$ as the randomness for an extractable commitment scheme.
   (d) It samples $t_i \leftarrow \{0,1\}^*$ as the randomness for generating a garbled circuit.
   (e) It computes $\mathsf{Com}^i_1 := \mathsf{ECom}_1(1^\lambda, (y_i, s_i, t_i, \{\mathsf{lab}^{i,j}_b, \overline{\mathsf{lab}}^{i,j}_b\}_{j\in[k],b\in\{0,1\}}); r_i)$.

3. For each $i \in [m]$ and $j \in [k]$,
   (a) It computes $\mathsf{otm}_1^{i,j} \leftarrow \mathsf{OT}_1(1^\lambda, (\mathsf{lab}_0^{i,j}, \mathsf{lab}_1^{i,j}); s_{i,j})$
4. It sets msg_1 to be $\{\mathsf{Com}_1^i\}_{i\in[m]}, \{\mathsf{otm}_1^{i,j}\}_{i\in[m],j\in[k]}$ and sets st to be its random tape.

Description of Ext. Ext does the following:

1. Ext runs $\mathsf{Ext}_{\mathsf{OT}}$ on $\mathsf{otm}_2^{i,j}$ for each $i \in [m]$ and $j \in [k]$ to compute $x_{i,j}$. It also runs $\mathsf{Ext}_{\mathsf{OT}^{(k,m)}}$ on otm to compute K.
2. It starts running Sim_Φ by corrupting the receiver client and the set of servers indexed by K. It provides to Sim_Φ with $\{y_i\}_{i\in K}$ as the first round dummy messages generated on behalf of the honest client. It provides $\{x_i\}_{i\notin K}$ as the first round message sent by the corrupt receiver client to the honest servers.
3. Sim_Φ queries the ideal functionality on input x.
4. Ext outputs x and sets st' to be its random tape along with K.

Description of Sim_R^2. Sim_R^2 does the following:

1. For each $i \in [m]$, it generates Com_3^i honestly as described in the protocol.
2. For each $i \in K$, it generates $\widetilde{\Phi}_{2,i}$ honestly as described in the protocol.
3. It starts running Sim_Φ using the random tape output in st'' and receives the second round message $\{\phi_i\}_{i\notin K}$ output by the honest servers.
4. For each $i \notin K$, it generates $\widetilde{\Phi}_{2,i}$ as $\mathsf{Sim}_{\mathsf{GC}}(1^\lambda, 1^{|\Phi_{2,i}|}, \{\mathsf{lab}_{x_{i,j}}^{i,j}, \overline{\mathsf{lab}}^{i,j}\}_{j\in[k]}, \phi_i)$ where $\overline{\mathsf{lab}}^{i,j}$ is uniformly chosen.
5. It generates $\mathsf{otm}_2 \leftarrow \mathsf{OT}_2^{(\lambda,m)}(\mathsf{otm}_1, (r_1^*, \ldots, r_m^*))$ where $r_i^* = r_i$ for each $i \in K$ and is $\perp$ otherwise.
6. It sets msg_3 to be $\{\mathsf{Com}_3^i, \widetilde{\Phi}_{2,i}\}_{i\in[m]}, \{\mathsf{otm}_3^{i,j}, \overline{\mathsf{lab}}^{i,j}\}_{i\in[m],j\in[k]}, \mathsf{otm}_2$ where $\overline{\mathsf{lab}}^{i,j} = \overline{\mathsf{lab}}_{y_{i,j}}^{i,j}$ for each $i \in K$ and $j \in [k]$.

Proof of Indistinguishability.

- $\underline{\mathsf{Hyb}_0}$: This hybrid corresponds to $\mathsf{View}_{\mathcal{A}}(\langle \mathcal{A}(1^\lambda), S(1^\lambda, y)\rangle)$.
- $\underline{\mathsf{Hyb}_1}$: In this hybrid, for each $i \in [m]$ and $j \in [k]$, we run $\mathsf{Ext}_{\mathsf{OT}}(\mathsf{otm}_2^{i,j})$ to obtain $x_{i,j}$ and $\mathsf{Ext}_{\mathsf{OT}^{(\lambda,m)}}(\mathsf{otm}_1)$ to obtain K. We generate $\mathsf{otm}_2 \leftarrow \mathsf{OT}_2^{(\lambda,m)}(\mathsf{otm}_1, (r_1^*, \ldots, r_m^*))$ where $r_i^* = r_i$ for each $i \in K$ and is $\perp$ otherwise. Via an identical argument given in Lemma 4.10, we can show that Hyb_0 and Hyb_1 are computationally indistinguishable.
- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we make the following changes:
  1. We repeat the following for $\lambda \cdot \binom{m}{\lambda}$ iterations:

(a) In each iteration, we randomly choose a set $K' \subseteq [m]$ of size λ.

(b) When we obtain the second round message from $\mathcal{A}$, we run $\mathsf{Ext}_{\mathsf{OT}^{(\lambda,m)}}(\mathsf{otm}_1)$ to obtain K.

(c) If $K \neq K'$, we move to the next iteration and if $K = K'$, we proceed as before.

2. If we are unable to proceed in each of the $\lambda \cdot \binom{m}{\lambda}$ iterations, we output a special symbol fail.

Via an identical argument to Lemma 4.11, we can show that Hyb_1 and Hyb_2 are statistically close.

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

  1. We repeat the following for $\lambda \cdot \binom{m}{\lambda}$ iterations:

     (a) We randomly choose a set $K' \subseteq [m]$ of size λ.

     (b) For each $i \notin K'$, we generate Com_1^i as the first round message of an extractable commitment to a dummy message. For each $i \in K'$, we generate Com_1^i as in the previous hybrid.

     (c) When we obtain the second round message from $\mathcal{A}$, we run $\mathsf{Ext}_{\mathsf{OT}^{(\lambda,m)}}(\mathsf{otm}_1)$ to obtain K.

     (d) If $K \neq K'$, we move to the next iteration. If $K = K'$, then for each $i \notin K'$, we generate Com_3^i as the third round message for an extractable commitment to a dummy message and for each $i \in K'$, we generate Com_3^i as in the previous hybrid.

  Via an identical argument to Lemma 4.12, we can show that $\mathsf{Hyb}_2 \approx_{\epsilon \cdot \lambda \cdot \binom{m}{\lambda} \cdot m} \mathsf{Hyb}_3$.

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

  1. We repeat the following for $\lambda \cdot \binom{m}{\lambda}$ iterations:

     (a) We randomly choose a set $K' \subseteq [m]$ of size λ.

     (b) For each $i \notin K'$ and for every $j \in [k]$, we generate $(\mathsf{otm}_1^{i,j}, \mathsf{st}^{i,j}) \leftarrow \mathsf{Sim}^1_{R,\mathsf{OT}}(1^\lambda)$. For every $i \in K'$ and for every $j \in [k]$, we generate $\mathsf{otm}_1^{i,j}$ as in the previous hybrid.

     (c) When we obtain the second round message from $\mathcal{A}$, we run $\mathsf{Ext}_{\mathsf{OT}^{(\lambda,m)}}(\mathsf{otm}_1)$ to obtain K and for each $i \in [m]$ and $j \in [k]$, $\mathsf{Ext}_{\mathsf{OT}}(\mathsf{otm}_1^{i,j})$ to obtain $x_{i,j}$.

     (d) If $K \neq K'$, we move to the next iteration. If $K = K'$, then for each $i \notin K'$ and $j \in [k]$, we generate $\mathsf{otm}_3^{i,j} \leftarrow \mathsf{Sim}^2_{R,\mathsf{OT}}(\mathsf{st}^{i,j}, \mathsf{lab}^{i,j}_{x_{i,j}})$ and for each $i \in K'$ and $j \in [k]$, we generate $\mathsf{otm}_3^{i,j}$ as in the previous hybrid.

  In Lemma 5.4, we show that $\mathsf{Hyb}_3 \approx_{\delta \cdot \lambda \cdot \binom{m}{\lambda} \cdot m} \mathsf{Hyb}_4$.

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

  1. In the iteration where $K = K'$, for each $i \notin K'$, we generate $\widetilde{\Phi}_{2,i}$ as $\mathsf{Sim}_{\mathsf{GC}}(1^\lambda, 1^{|\Phi_{2,i}|}, \{\mathsf{lab}^{i,j}_{x_{i,j}}, \overline{\mathsf{lab}}^{i,j}_{y_{i,j}}\}_{j \in [k]}, \phi_i)$ where $\phi_i := \Phi_{2,i}(x_i, y_i)$.

  In Lemma 5.5, we show that $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$.

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

    1. We repeat the following for $\lambda \cdot \binom{m}{\lambda}$ iterations:
        (a) We randomly choose a set $K' \subseteq [m]$ of size λ.
        (b) We run the simulator Sim_Φ for the protocol Φ by corrupting the receiver client and the set of servers indexed by K'. We obtain $\{y_i\}_{i \in K'}$ as the first round message from the honest sender client to the corrupted servers. We use this to generate the first round message of the protocol as in the previous hybrid.
        (c) When we obtain the second round message from $\mathcal{A}$, we run $\mathsf{Ext}_{\mathsf{OT}^{(\lambda,m)}}(\mathsf{otm}_1)$ to obtain K and for each $i \in [m]$ and $j \in [k]$, $\mathsf{Ext}_{\mathsf{OT}}(\mathsf{otm}_1^{i,j})$ to obtain $x_{i,j}$.
        (d) If $K \neq K'$, we go to the next iteration. Else, if $K = K'$, we send $\{x_i\}_{i \notin K'}$ as the first round message from the corrupted receiver to the honest servers. When Sim_Φ makes a query to the ideal functionality with input x, we reply with $f(x, y)$.
        (e) Sim_Φ sends $\{\phi_i\}_{i \notin K'}$ as the second round message from the honest servers to the corrupt client. We use this to generate $\{\widetilde{\Phi}_{2,i}\}_{i \notin C}$.

    In Lemma 5.6, we show that $\mathsf{Hyb}_5 \approx_{\mu \cdot \lambda \cdot \binom{m}{\lambda}} \mathsf{Hyb}_6$.

- $\underline{\mathsf{Hyb}_7}$: In this hybrid, we reverse the changes made in Hyb_4. Specifically, in each of the iterations, instead of generating $\{\mathsf{otm}_1^{i,j}, \mathsf{otm}_3^{i,j}\}_{i \notin K', j \in [k]}$ using the simulator $\mathsf{Sim}^1_{R,\mathsf{OT}}$, $\mathsf{Sim}^2_{R,\mathsf{OT}}$ respectively, we generate them using the actual algorithms OT_1 and OT_3 respectively on input $\{\mathsf{lab}_0^{i,j}, \mathsf{lab}_1^{i,j}\}$. It now follows that $\mathsf{Hyb}_6 \approx_{\delta \cdot \lambda \cdot \binom{m}{\lambda} \cdot m} \mathsf{Hyb}_7$ via an identical argument to the proof of indistinguishability between Hyb_3 and Hyb_4.

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

    1. We repeat the following for $\lambda \cdot \binom{m}{\lambda}$ iterations:
        (a) We randomly choose a set $K' \subseteq [m]$ of size λ.
        (b) Instead of generating $\{y_i\}_{i \in K'}$ as the output of $\mathsf{Sim}_{\mathsf{OT}}$, we compute it as follows: we generate $(y_1, \ldots, y_m) \leftarrow \Phi_1(\mathbf{0})$ (where $\mathbf{0}$ is the default input) and then output $\{y_i\}_{i \in K'}$.

    It follows from the perfect first message indistinguishability of the protocol Φ, we have $\mathsf{Hyb}_7 \equiv \mathsf{Hyb}_8$.

- $\underline{\mathsf{Hyb}_9}$: In this hybrid, we reverse the changes made in Hyb_3. Specifically, in each iteration, instead of generating $\{\mathsf{Com}_1^i, \mathsf{Com}_3^i\}_{i \notin K'}$ as commitments to some dummy message, we generate them as commitment to $(y_i, s_i, t_i, \{\mathsf{lab}_{i,j,b}, \overline{\mathsf{lab}}_{i,j,b}\}_{j \in [k], b \in \{0,1\}})$ where y_i is the message to the i-th server when the input of the sender is $\mathbf{0}$, $s_i := (s_{i,1}, \ldots, s_{i,k})$ is such that $s_{i,j}$ is the randomness used in generating $\{\mathsf{otm}_1^{i,j}, \mathsf{otm}_1^{i,j}\}$ for each $j \in [k]$ and t_i is an uniformly chosen random string from $\{0,1\}^*$. Via an identical argument proving indistinguishability between Hyb_2 and Hyb_3, we can show that $\mathsf{Hyb}_8 \approx_{\epsilon \cdot \lambda \cdot \binom{m}{\lambda} \cdot m} \mathsf{Hyb}_9$.

- $\underline{\mathsf{Hyb}_{10}}$: In this hybrid, we reverse the changes made in Hyb_2. It now follows via a similar argument in proving statistical indistinguishability between Hyb_1 and Hyb_2, we can show that

$\mathsf{Hyb}_9 \approx_s \mathsf{Hyb}_{10}$. Note that Hyb_{10} is identically distributed to the ideal world output generated by Sim_R.

Lemma 5.4. *Assuming the (T_2, δ)-security of the oblivious transfer protocol against malicious receivers, we have* $\mathsf{Hyb}_3 \approx_{\delta \cdot \lambda \cdot \binom{m}{\lambda} \cdot m}$.

Proof. We consider each rewinding and show that in each of these rewindings, the changes described in Hyb_4 cannot be distinguished from Hyb_3 except with $m \cdot \delta$ advantage. We show this for the first rewinding thread and the case for any general rewinding thread is identical.

Assume for the sake of contradiction that the above change made to the first rewinding thread can be distinguished with advantage at least $m \cdot \delta$. By a standard averaging argument, there exists two intermediate hybrids $\mathsf{Hyb}_{3,i}, \mathsf{Hyb}_{3,i-1}$ (described below) such that they are computationally distinguishable with advantage δ. We now give the description of the two hybrids. In both hybrids, for every $i' < i$ and if $i' \notin K'$, we generate $\{\mathsf{otm}_1^{i',j}, \mathsf{otm}_3^{i',j}\}_{j \in [k]}$ as the output of the simulator $\mathsf{Sim}_{R,\mathsf{OT}}$. In both hybrids, for every $j' > j$ or if $j' < j$ and $j' \in K'$, we generate $\{\mathsf{otm}_1^{i',j}, \mathsf{otm}_3^{i',j}\}$ for each $j \in [k]$ as the output of $\mathsf{OT}_1, \mathsf{OT}_3$ respectively on input $\{\mathsf{lab}_0^{i',j}, \mathsf{lab}_1^{i',j}\}$. The only difference between these two hybrids in the generation of $\{\mathsf{otm}_1^{i,j}, \mathsf{otm}_3^{i,j}\}_{j \in [k]}$. Specifically, if $i \notin K'$, we generate it as the output of the simulator in $\mathsf{Hyb}_{3,i}$ and in $\mathsf{Hyb}_{3,i-1}$, we generate it using $\mathsf{OT}_1, \mathsf{OT}_3$ on input $\{\mathsf{lab}_0^{i,j}, \mathsf{lab}_1^{i,j}\}$. We now use a distinguisher against $\mathsf{Hyb}_{3,i}$ and $\mathsf{Hyb}_{3,i-1}$ to contradict the (T_2, δ)-security against malicious receivers. Note that if $i \in K'$, then $\mathsf{Hyb}_{3,i}$ and $\mathsf{Hyb}_{3,i-1}$ are identically distributed. Hence, in the rest of the proof, we assume w.l.o.g. that $i \notin K'$.

We interact with the external challenger and provide $\{\mathsf{lab}_0^{i,j}, \mathsf{lab}_1^{i,j}\}_{j \in [k]}$ as the set of sender inputs. We define an adversary $\mathcal{A}'$ that receives $\{\mathsf{otm}_1^{i,j}\}_{j \in [k]}$ externally, and generates the rest of the first round protocol messages as in $\mathsf{Hyb}_{3,i-1}$. It runs $\mathcal{A}$ internally on the first round message and receives the second round message. When it receives the second round message from $\mathcal{A}$, it forwards $\{\mathsf{otm}_2^{i,j}\}_{j \in [k]}$ externally. It receives $\{\mathsf{otm}_3^{i,j}\}_{j \in [k]}$ externally and computes the rest of the third round messages as in $\mathsf{Hyb}_{3,i-1}$. We interact with the external challenger using the adversary $\mathcal{A}'$. We finally obtain the view of $\mathcal{A}'$ from the challenger and use it to extract the view of $\mathcal{A}$ and output it.

Note that if the messages generated by the challenger are using the algorithms $\mathsf{OT}_1, \mathsf{OT}_3$ then the output of the reduction is identically distributed to $\mathsf{Hyb}_{3,i-1}$ Else, it is identically distributed to $\mathsf{Hyb}_{3,i}$. Furthermore, the running time of the reduction is $T_1 \cdot \mathsf{poly}(\lambda \cdot \binom{m}{\lambda}) \leq T_2$. Hence, if $\mathsf{Hyb}_{3,i-1}$ and $\mathsf{Hyb}_{3,i}$ can be distinguished with advantage δ, then the above reduction breaks the (T_2, δ)-security of OT against malicious receivers and this is a contradiction. □

Lemma 5.5. *Assuming the security of garbled circuits, we have* $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$.

Proof. Assume for the sake of contradiction that Hyb_4 and Hyb_5 are computationally distinguishable. We now give a reduction to the security of garbled circuits.

We interact with the garbled circuits challenger and give for each $i \in K'$, $\Phi_{2,i}$ as the challenge circuit, (x_i, y_i) as the challenge inputs and $\{\mathsf{lab}_{x_{i,j}}^{i,j}, \mathsf{lab}_{y_{i,j}}^{i,j}\}_{j \in [k]}$ as the challenge input labels. We obtain $\{\widetilde{\Phi}_{2,i}\}_{i \notin K'}$ from the external challenger and use it to generate the third round message as in Hyb_4. We finally output the view of $\mathcal{A}$.

Note that if the garbled circuits $\{\widetilde{\Phi}_{2,i}\}_{i \notin K'}$ were generated by the external challenger as the output of the simulator $\mathsf{Sim}_{\mathsf{GC}}$ then the output of the reduction is identically distributed to Hyb_5.

Otherwise, it is identically distributed to Hyb_4. Thus, if Hyb_4 and Hyb_5 are computationally distinguishable then the above reduction breaks the security of garbled circuits and this is a contradiction. $\square$

Lemma 5.6. *Assuming μ-statistical security of the protocol Φ, we have that* $\mathsf{Hyb}_5 \approx_{\mu\cdot\lambda\cdot\binom{m}{\lambda}} \mathsf{Hyb}_6$.

Proof. We consider each rewinding and show that in each of these rewindings, the changes described in Hyb_5 cannot be distinguished from Hyb_6 except with μ advantage. We show this for the first rewinding thread and the case for any general rewinding thread is identical.

Assume for the sake of contradiction that the above change made to the first rewinding thread can be distinguished with advantage at least μ. We give a reduction to the security of the protocol Φ.

We interact with the external challenger by giving y as the input of the sender. We corrupt the set of servers indexed by set K' and obtain $\{y_i\}_{i\in K'}$. We use it to generate the first round message of the protocol and send it to $\mathcal{A}$. When we receive the second round message from $\mathcal{A}$, for each $i \in [m]$ and $j \in [k]$, we run $\mathsf{Ext}_{\mathsf{OT}}(\mathsf{otm}_2^{i,j})$ and obtain $x_{i,j}$ and $\mathsf{Ext}_{\mathsf{OT}^{(\lambda,m)}}(\mathsf{otm}_1)$ to obtain K. If $K \neq K'$, we move to the next iterations and proceed as before. Otherwise, we send $\{x_i\}_{i\notin K'}$ as the first round message from the corrupt receiver to the honest servers. We obtain $\{\phi_i\}_{i\notin K'}$ from the external challenger. We use it to generate the last round message of the protocol and finally output the view of $\mathcal{A}$.

Note that if the messages of the protocol Φ are generated by the external challenger by running Sim_Φ then the output of the above reduction is identically distributed to Hyb_6. Otherwise, it is distributed identically to Hyb_5. Thus, if Hyb_5 and Hyb_6 are statistically distinguishable with advantage μ, then the above reduction breaks the μ-security of the protocol Φ and this is a contradiction. $\square$

5.5 Additional Properties

We now describe how to add certain additional properties that are needed for our applications.

k-rewinding Sender Security. For our application, we need a protocol that is secure against a malicious receiver that rewinds the honest sender k times (for an a priori bounded constant k) by giving potentially different second round messages and requiring to compute possibly different functionalities $f_1, \ldots, f_k$. For the construction described in Figure 3 to have this property, we need the building blocks to have the following properties:

- **k-rewinding secure Extractable Commitment:** This can be instantiated using the construction given in [BGJ+18].
- **k-rewinding secure three-round OT protocol:** This can be instantiated using the modifications described in Section 4.5.
- **Special Outer Protocol:** We need an outer protocol where the first round message sent by the clients could be reused k times to compute possibly k different functionalities. [IKSS21] showed how to modify the protocol given in [IKP10, Pas12] to be reused 2 times. We now show to modify this for arbitrary constant k number of reuses. Note that the first round message from the client in the IKP protocol [IKP10, Pas12] 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 the protocol to be reused k times, we make use of perfect hash functions [FK84]. Recall that a perfect hash function family $\{h_i\}_{i\in[\ell]}$ is a set of ℓ functions each mapping $\{0,1\}^n \to [k]$ such that for any subset $S \subseteq \{0,1\}^n$ of size k, there exists at least one $i \in [\ell]$ such that h_i is injective on S. It is possible to show that if we fix $\ell = O(n+\lambda)$, then with overwhelming probability ℓ k-wise independent hash functions form a perfect hash family (see [BS19, Lemma 4.5]). In our protocol, we let the sender choose ℓ k-wise independent functions and send this to the receiver in the first round (with overwhelming probability, this will be a perfect hash function family). For the first round message in the IKP protocol to be reused k times, we generate $k \times \ell$ of the MCDS messages and random secret sharings of 0 in the first round. The servers arrange these messages in a matrix of k rows and ℓ columns. In the second round, the servers compute $h_i(f)$ for each $i \in [\ell]$ and they use MCDS message and the secret sharing of 0 corresponding to $h_i(f)$-th row for each column $i \in [\ell]$. 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 for distinct $f_1, \ldots, f_k$, there exists at least one index $i \in [\ell]$ such that the MCDS message and the random secret sharing of 0 is only used once in the k executions. We note that this modification still preserves the perfect first message indistinguishability and delayed functions selection.

Apart from these, we also change the protocol such that otm_1 is sent in the first round (as we described for the OT case in Section 4.5).

Public-Coin Second Round message. Note that with the above mentioned changes, the second round message sent by the receiver consists of public coins.

Two Properties of Sim_S.

1. We note that the view of the adversary generated by Sim_S is identically distributed to its view when interacting with an honest receiver with default input x'. This property is directly inherited from $\mathsf{Sim}_S^{\mathsf{OT}}$.

2. We note that to extract the sender input y, Sim_S rewinds and extracts from the extractable commitment. Thus, for a given first round message from the malicious sender and valid third round responses to two random second round challenges, Sim_S can use the above information to extract (M_0, M_1) except with negligible probability. If the extractable commitment on the other hand, requires t accepting transcripts (with valid third round messages), then Sim_S also needs t accepting transcripts with valid third round messages. Additionally, if $\mathsf{Sim}_S^{\mathsf{OT}}$ requires only two accepting transcripts to extract $\{\mathsf{lab}_0^{i,j}, \mathsf{lab}_1^{i,j}\}_{i\in[m],j\in[k]}$ then so does Sim_S.

Existence of Straight-Line SPS extractor against Malicious Senders. We note that Sim_S described earlier rewinds the extractable commitment as well as the OT protocol to extract the sender inputs in the 2PC protocol and to generate the output of the honest receiver. If the extractable commitment is straight-line extractable in super-polynomial time (such a construction of extractable commitment is given in Appendix A) and if the OT protocol also admits a straight-line

SPS extractor (we describe how to add this property to our OT protocol in Section 4.5), we get a super-polynomial time, straight-line simulator against malicious senders.

6 3-Round Two-Party Computation with Special Extraction

In this section, we give a construction of a three-round, two-party protocol for computing NC^1 functionalities that satisfies certain special properties. We then use this special two-party protocol as the key building block in constructing a protocol that implements the watchlist functionality. In Section 6.1, we give the syntax this two party protocol and formally define the properties that it needs to satisfy. In Section 6.2, we give the description of the building blocks needed in the construction. In Section 6.3, we give the construction of the protocol and in Section 6.4, we give the proof of security. In Section 6.6, we describe how to add certain additional properties to this protocol (which are needed to construct a watchlist protocol).

6.1 Definition

We start with the syntax of the protocol.

Syntax. A three-round protocol $\Pi = (\Pi_1, \Pi_2, \Pi_3, \mathsf{out}_\Pi)$ between a sender and a receiver proceeds as follows. In each round $r \in [3]$, the sender runs Π_r on its identity, the transcript, its input and randomness to generate msg_r^S. Similarly, in round r, the receiver runs Π_r on its identity, the transcript, its input and randomness to generate msg_r^R. The sender sends msg_r^S to the receiver and the receiver sends msg_r^R to the sender and these messages are then added to the transcript. At the end of the protocol, the receiver run out_Π on its identity, transcript, its input and randomness to compute the output which is a string z or $\bot$. The sender runs out_Π on its identity and the first round message from the receiver, the second round message from the sender and third round message from the receiver and outputs either $\mathsf{accept}/\mathsf{reject}$. We note that while the output computation of the receiver requires access to its private random tape, the output of the sender is publicly computable.

Definition 6.1. *A three-round two-party protocol* $\Pi = (\Pi_1, \Pi_2, \Pi_3, \mathsf{out}_\Pi)$ *for computing a function* f *is said to satisfy* k*-special extraction if:*

- ***Public Coin Second Round Messages.*** *The second round messages from the sender and the receiver are both public coin.*
- ***Security against Malicious Senders.*** *There exists an expected PPT machine* Sim_S *such that for every non-uniform* $\mathcal{A}$ *the corrupts the sender and for every receiver's input* $x \in \{0,1\}^n$*, we have:*

$$\Big\{\big(\mathsf{View}_{\mathcal{A}}(\langle R(1^\lambda, x), \mathcal{A}(1^\lambda)\rangle), \mathsf{out}_R(\langle R(1^\lambda, x), \mathcal{A}(1^\lambda)\rangle)\big)\Big\} \approx_c$$
$$\Big\{(\mathsf{View}_{\mathcal{A}}, f(x,y)) : (\mathsf{View}_{\mathcal{A}}, y) \leftarrow (\mathsf{Sim}_S)^{\mathcal{A}}(1^\lambda)\Big\}$$

In the above definition, we note that if y *output by* Sim_S *is the special symbol* $\bot$*, then the output of* f *is also* $\bot$*.*

- Run $\mathsf{SPSim}^1_R(1^\lambda)$ to obtain $(\mathsf{msg}^S_1, \mathsf{st})$ and send msg^S_1 to $\mathcal{A}$. Receive msg^R_1 from $\mathcal{A}$.
- Run $\mathsf{SPExt}_R(\mathsf{msg}^R_1)$ to obtain x, st'.
- Sample $(\mathsf{msg}^S_2, \mathsf{st}'')$ from $\mathsf{SPSim}^2_R(\mathsf{st}, \mathsf{st}')$ and send msg^S_2 to $\mathcal{A}$.
- Receive msg^R_2 from $\mathcal{A}$.
- Run $\mathsf{SPSim}^3_R(\mathsf{st}'', f(x,y), \mathsf{msg}^R_1, \mathsf{msg}^R_2, \mathsf{msg}^S_2)$ to obtain msg^S_3 and send this to $\mathcal{A}$.
- Receive msg^R_3 from $\mathcal{A}$.
- Output view of $\mathcal{A}$.

Figure 4: Description of Ideal_R.

- ***Super-Polynomial Time Simulation Security against Malicious Receivers.*** *There exists a super-polynomial time machine* $\mathsf{SPSim}_R = (\mathsf{SPSim}^1_R, \mathsf{SPExt}_R, \mathsf{SPSim}^2_R, \mathsf{SPSim}^3_R)$ *such that for every adversary* $\mathcal{A}$ *corrupting the receiver and for every sender's input* $y \in \{0,1\}^n$*, we have:*

$$\left\{\mathsf{View}_{\mathcal{A}}(\langle \mathcal{A}(1^\lambda), S(1^\lambda, y)\rangle)\right\} \approx_c \mathsf{Ideal}_R(1^\lambda, y, \mathcal{A}, \mathsf{SPSim}_R)$$

  where the experiment Ideal_R *is described in Figure 4.*

- ***Special Extraction of the Malicious Receiver Input.*** *There exists a super-polynomial time extractor* $\mathsf{SPSpecExt}_R$ *such that for any adversary* $\mathcal{A}$ *corrupting the receiver and for any sender input* $y \in \{0,1\}^n$*, the probability the following experiment outputs 1 is negligible:*

  1. *Sample a transcript* $\mathbb{T}$ *from* $\mathsf{Ideal}_R(1^\lambda, y, \mathcal{A}, \mathsf{SPSim}_R)$.
  2. *If the output of the sender* S *in the transcript* $\mathbb{T}$ *is* reject*, then output of the experiment is* 0.
  3. *Run* $\mathsf{SPExt}_R(\mathsf{msg}^1_R)$ *(where* $\mathsf{msg}^1_R \in \mathbb{T}$*) to obtain* x*. If* $x = \perp$*, output of the experiment is* 0.
  4. *Else, run* $\mathsf{SPSpecExt}(\mathbb{T})$ *to obtain* x'.
  5. *The output of the experiment is* 1 *if and only if* $x \neq x'$.

- ***Existence of*** k ***accepting Transcript Extractor.*** *There exists a polynomial time machine* Ext_R *that on input any* k *transcripts* $\mathbb{T}_1, \ldots, \mathbb{T}_k$ *such that in each of the transcript the output of the sender is* accept *outputs* $\overline{x}$ *such that* $\overline{x} = \mathsf{SPSpecExt}(\mathbb{T}_1)$ *with overwhelming probability.*

6.2 Building Blocks

We now describe the building blocks used in the construction.

- **Outer MPC Protocol.** A two-round two clients, m-server MPC protocol $(\Phi_1, \Phi_2, \mathsf{out}_\Phi)$ for computing NC^1 functionalities that satisfies μ-statistical security with selective abort against adversary corrupting upto t servers and one of the clients. We set $t = \lambda$ and $m = 4t+1$. We additionally need this protocol to satisfy a couple of properties:

1. Perfect first round message indistinguishability, meaning that for any input of the honest client, the distribution of the first round message sent to the corrupted servers from the honest client is identically distributed to the first round message generated by the simulator Sim_Φ on behalf of this client.
2. The parties can choose a set of $3t+1$ servers among the m servers to receive the second round message from and compute the output of the functionality from these messages.
3. Let $\mathcal{A}$ be an adversary that corrupts one of the clients and at most t of the m servers. Let $\{x_i\}_{i \in H}$ be the first round message sent by $\mathcal{A}$ to the honest servers H where $|H| \geq 3t+1$. There is a polynomial time algorithm Rec such that for any $C \subset H$ of size at most t, $\mathsf{Rec}(C, \{x_i\}_{i \notin C})$ is the same as the query x made by Sim_Φ to its ideal functionality as long as $x \neq \bot$

We call this protocol as the outer protocol. We note that the protocol given in [IKP10, Pas12] satisfies all these properties. The first two properties are direct consequence of their construction and the third property follows from the error correction property of pairwise verifiable secret sharing used in their construction.

- **Inner Protocol.** For each $i \in [m]$, we use a three-round inner protocol $(\Pi_{i,1}, \Pi_{i,2}, \Pi_{i,3}, \mathsf{out}_\Pi)$ for computing the functionality $\Phi_2(i, \cdot)$ (i.e., the computation done by the i-th server) that satisfies (T_2, ϵ)-security against malicious senders and super-polynomial simulation security against malicious receivers with the two additional properties of Sim_S described in Section 5.5 and the public-coin second round message from the receiver. Let t be the number of accepting transcripts (for a constant t) required by Sim_S needed to extract y.
- A t-rewinding (T_2, ϵ) secure, straight-line SPS extractable commitment scheme $\mathsf{ECom} = (\mathsf{ECom}_1, \mathsf{ECom}_2, \mathsf{ECom}_3)$ constructed in Appendix A.

Setting the Parameters. We set $T_2 \geq \mathsf{poly}(\binom{m}{k} \cdot \lambda)$ and $\epsilon = \mu$ such that $\epsilon \cdot \binom{m}{k} \leq \mathsf{negl}(\lambda)$.

6.3 Construction

We describe the protocol in Figure 5.

6.4 Proof of Security

In this subsection, we show that the construction given in Figure 5 satisfies Definition 6.1. The public coin second round message property follows from the observation that $K, \{\mathsf{Com}_{i,2}\}_{i \in [m]}$ chosen by the sender and $\{\pi_{i,2}\}_{i \in [m]}$ chosen by the receiver are both public coin.

6.4.1 Security against Malicious Senders

Description of Sim_S. Sim_S does the following:

- It computes $(x_1, \ldots, x_m) \leftarrow \phi_1(\mathbf{0})$ (where $\mathbf{0}$) is the default input.

- **Round-1:**
  - $\underline{S \to R:}$ S does the following:
    1. It computes $(y_1, \ldots, y_m) \leftarrow \Phi_1(y)$.
    2. For each $i \in [m]$,
       (a) It computes $\pi_{i,1}^S \leftarrow \Pi_{i,1}(1^\lambda, S, y_i)$.
    3. It sends $\{\pi_{i,1}^S\}_{i\in[m]}$ to R.
  - $\underline{R \to S}$: R does the following:
    1. It computes $(x_1, \ldots, x_m) \leftarrow \Phi_1(x)$.
    2. For each $i \in [m]$,
       (a) It chooses $r_i \leftarrow \{0,1\}^*$ as the receiver randomness used in the protocol Π_i.
       (b) It computes $\pi_{i,1}^R := \Pi_{i,1}(1^\lambda, R, x_i; r_i)$.
       (c) It chooses $s_i \leftarrow \{0,1\}^*$ as the randomness for an extractable commitment scheme.
       (d) It computes $\mathsf{Com}_{i,1} := \mathsf{ECom}_1(1^\lambda, (x_i, r_i); s_i)$
    3. It sends $\{\pi_{i,1}^R, \mathsf{Com}_{i,1}\}_{i\in[m]}$ to S.
- **Round-2:**
  - $\underline{S \to R:}$ S does the following:
    1. For each $i \in [m]$:
       (a) It chooses $\mathsf{Com}_{i,2} \leftarrow \mathsf{ECom}_2(\mathsf{Com}_{i,1})$.
    2. It chooses a random subset $K \subset [m]$ of size λ.
    3. It sends $K, \{\mathsf{Com}_{i,2}\}_{i\in[m]}$.
  - $\underline{R \to S}$: R does the following:
    1. For each $i \in [m]$,
       (a) It computes $\pi_{i,2} := \Pi_{i,2}(\pi_{i,1}^S)$.
    2. It sends $\{\pi_{i,2}\}_{i\in[m]}$ to S.
- **Round-3**:
  - $\underline{S \to R:}$ S does the following:
    1. For each $i \in [m] \setminus K$:
       (a) It computes $\pi_{i,3} \leftarrow \Pi_{i,3}(\{\pi_{i,1}^R, \pi_{i,2}\}, y_i)$.
    2. It sends $\{\pi_{i,3}\}_{i\in[m]\setminus K}$.
  - $\underline{R \to S}$: R does the following:
    1. For each $i \in [m]$,
       (a) It computes $\mathsf{Com}_{i,3} \leftarrow \mathsf{ECom}_3(\mathsf{Com}_{i,2}, (x_i, r_i); s_i)$.
    2. It sends $\{\mathsf{Com}_{i,3}\}_{i\in[m]}, \{s_i\}_{i\in K}$ to S.
- **Output Computation:**
  - $\underline{S}$: To compute the output, S does the following:
    1. For each $i \in [m]$, it runs $\mathsf{Verify}(\mathsf{Com}_1^i, \mathsf{Com}_2^i, \mathsf{Com}_3^i)$.
    2. For each $i \in K$:
       (a) It recovers (x_i, r_i) from $\mathsf{Com}_{i,1}$ using randomness s_i.
       (b) It checks if $\pi_{i,1}^R$ is consistent with (x_i, r_i).
    3. If any of the checks fail, it rejects. Otherwise, it accepts.
  - $\underline{R}$: To compute the output, R does the following:
    1. For each $i \in [m] \setminus K$, it computes $\phi_i := \mathsf{out}_{\Pi_i}(\pi_{i,1}^S, \pi_{i,3}, (x_i, r_i))$.
    2. It outputs $\mathsf{out}_\Phi(\{\phi_i\}_{i\in[m]\setminus K})$.

Figure 5: Descriptions of the protocol.

- It generates the messages in the inner protocol using $\mathsf{Sim}^i_S(1^\lambda, x_i)$ using random tape u_i for each $i \in [m]$.[10] It generates the messages of the i-th extractable commitment with input (x_i, u_i). It completes the execution with $\mathcal{A}$.
- For each $i \in [m] \setminus K$, it obtains y_i from Sim^i_S.
- It runs Sim_Φ by corrupting only the sender and the specifies the set of honest servers to be $[m] \setminus K$. It provides the first round message $\{y_i\}_{i \in [m] \setminus K}$ from the corrupt sender to the honest servers.
- Sim_Φ queries the ideal functionality on y and provides the instruction to either deliver the output to the receiver or output $\perp$. If the instruction is to output $\perp$, then Sim_S sets $y = \perp$.

Proof of Indistinguishability.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to $\big(\mathsf{View}_\mathcal{A}(\langle R(1^\lambda, x), \mathcal{A}(1^\lambda)\rangle), \mathsf{out}_R(\langle R(1^\lambda, x), \mathcal{A}(1^\lambda)\rangle)\big)$.
- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we make the following changes:
  1. We repeat the following for $\lambda \cdot \binom{m}{\lambda}$ iterations:
     (a) In each iteration, before sending the first round message on behalf of R, we choose a random subset $K' \subset [m]$ of size λ.
     (b) On receiving the second round message from $\mathcal{A}$, we check if $K' = K$.
     (c) If $K' = K$ then we proceed to the complete the execution as in the previous hybrid. Else, if $K' \neq K$, then we move to the next iteration.
  2. If we fail to complete the execution even after $\lambda \cdot \binom{m}{\lambda}$ trials, we output a special symbol fail and abort.

  Via an identical argument to the proof of Lemma 4.11, we can show that Hyb_1 and Hyb_0 are statistically close.
- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we make the following changes in each of the $\lambda \cdot \binom{m}{\lambda}$ iterations:
  1. In each iteration, before sending the first round message on behalf of R, we choose a random subset $K' \subset [m]$ of size λ.
  2. For each $i \notin K'$, we generate $\{\mathsf{Com}_{i,1}, \mathsf{Com}_{i,3}\}$ as extractable commitments to a dummy message.

  Assuming the (T_2, ϵ)-hiding property of the extractable commitment scheme ECom, using an identical argument to Lemma 4.12, we can show that $\mathsf{Hyb}_1 \approx_{\epsilon \cdot m \cdot \lambda \cdot \binom{m}{\lambda}} \mathsf{Hyb}_2$.
- $\underline{\mathsf{Hyb}_3}$: In this hybrid, we make the following changes in each of the $\lambda \cdot \binom{m}{\lambda}$ iterations:
  1. Before sending the first round message on behalf of R, we choose a random subset $K' \subset [m]$ of size λ.

[10] Note that from the first property satisfied by Sim^i_S in Section 5.5, it takes some default input used by the receiver. In this case, the default input corresponds to x_i generated as above.

2. For each $i \notin K'$, instead of generating the messages $(\pi^R_{i,1}, \pi_{i,2})$ and computing the output of Π_i using input x_i and randomness r_i, we run $\mathsf{Sim}^i_S(1^\lambda, 0^k)$ (where $k = |x_i|$) to obtain these messages along with y_i. We compute the output of R using the extracted $\{y_i\}_{i \in [m] \setminus K'}$.

In Lemma 6.2, we show that $\mathsf{Hyb}_3 \approx_{\epsilon \cdot m \cdot \lambda \cdot \binom{m}{\lambda}} \mathsf{Hyb}_2$.

- $\underline{\mathsf{Hyb}_4}$: In this hybrid, in each of the $\lambda \cdot \binom{m}{\lambda}$ iterations:

  1. Before sending the first round message on behalf of R, we choose a random subset $K' \subset [m]$ of size λ.
  2. We initialize the simulator Sim_Φ for the protocol Φ by corrupting the sender client and corrupt the set of servers indexed by K'. We obtain $\{x_i\}_{i \in K'}$ from Sim and use this to generate the first round message.
  3. In the iteration where $K = K'$, we send $\{y_i\}_{i \notin K}$ to Sim_Φ as the first round message from the corrupted sender client to the honest servers. Sim_Φ queries the ideal functionality on input y and we record this value.
  4. If Sim_Φ instructs the honest receiver to output $\bot$, we output $\bot$ and otherwise, we instruct the honest receiver to output $f(x, y)$.

  In Lemma 6.3, we show that $\mathsf{Hyb}_4 \approx_{\mu \cdot \lambda \cdot \binom{m}{\lambda}} \mathsf{Hyb}_3$.

- $\underline{\mathsf{Hyb}_5}$: In this hybrid, we make the following changes to each of the $\lambda \cdot \binom{m}{\lambda}$ iterations:

  1. Instead of generating $\{x_i\}_{i \in K'}$ using Sim_Φ, we generate it as follows. We compute $(x_1, \ldots, x_m) \leftarrow \Phi_1(\mathbf{0})$ (where $\mathbf{0}$ is the default input) and output $\{x_i\}_{i \in K'}$.

  It follows from perfect indistinguishability of the first round message of the protocol Φ that Hyb_4 and Hyb_5 are identically distributed.

- $\underline{\mathsf{Hyb}_6}$: In this hybrid, we make the following changes in each of the $\lambda \cdot \binom{m}{\lambda}$ iterations:

  1. We compute $(x_1, \ldots, x_m) \leftarrow \Phi_1(\mathbf{0})$ and for each $i \notin K'$, we run $\mathsf{Sim}^i_S(1^\lambda, x_i)$ instead of $\mathsf{Sim}^i_S(1^\lambda, 0^k)$ to compute $\pi^R_{i,1}, \pi_{i,2}$ and y_i.

  Since $\mathsf{Sim}^i_S(1^\lambda, x) \approx_\epsilon \mathsf{Sim}^i_S(1^\lambda, x')$ for any x, x', it now follows that $\mathsf{Hyb}_6 \approx_{\epsilon \cdot m \cdot \lambda \cdot \binom{m}{\lambda}} \mathsf{Hyb}_5$.

- $\underline{\mathsf{Hyb}_7}$: In this hybrid, we reverse the changes made in Hyb_2. Specifically, in each of the $\lambda \cdot \binom{m}{\lambda}$ iterations, we compute $(x_1, \ldots, x_m) \leftarrow \Phi_1(\mathbf{0})$ and generate $\{\mathsf{Com}^i_1, \mathsf{Com}^i_3\}$ for each $i \notin K'$ as extractable commitments to (x_i, u_i).

  In Lemma 6.4, we show that $\mathsf{Hyb}_7 \approx_{\epsilon \cdot m \cdot \lambda \cdot \binom{m}{\lambda}} \mathsf{Hyb}_6$.

- $\underline{\mathsf{Hyb}_8}$: In this hybrid, we reverse the changes made in Hyb_1. Again, via an identical argument given above, we can show that $\mathsf{Hyb}_8 \approx_s \mathsf{Hyb}_7$. The output of Hyb_8 is identically distributed to the output of the ideal world execution using Sim_S.

Lemma 6.2. *Assuming that Π_i (for each $i \in [m]$) is ϵ-simulation secure against malicious senders running in time T_2, we have that* $\mathsf{Hyb}_3 \approx_{\epsilon \cdot \lambda \cdot \binom{m}{\lambda}} \mathsf{Hyb}_2$.

Proof. We consider each iteration and show that in each of these iterations, the changes described in Hyb_3 cannot be distinguished from Hyb_2 except with $\epsilon \cdot m$ advantage. We show this for the first iteration and the case for any general iteration is identical. Assume for the sake of contradiction that the changes made in the first iteration are distinguishable with advantage greater than ϵm. We now give a reduction that breaks the $\epsilon \cdot m$-simulation security of m instances of the inner protocol against malicious senders that run in time T_2. This directly contradicts the ϵ simulation security of a single instance (by a standard averaging argument).

The reduction chooses a random subset $K' \subset [m]$ of size λ and computes $(x_1, \ldots, x_m) \leftarrow \Phi_1(x)$. It provides $\{x_i\}_{i \notin K'}$ as the challenge receiver inputs. The reduction defines an adversary $\mathcal{A}'$ that internally runs $\mathcal{A}$ and interacts with the external challenger as follows. $\mathcal{A}'$ obtains $\{\pi^R_{i,1}\}_{i \notin K'}$ from the challenger and uses it to compute the first round message in the protocol. It receives the first round message from $\mathcal{A}$ and forwards $\{\pi^S_{i,1}\}_{i \notin K'}$ to the challenger. It obtains $\{\pi_{i,2}\}_{i \notin K'}$ from the challenger and uses this to generate the second round message of the protocol. It obtains the second round message from $\mathcal{A}$ and checks if $K' = K$. If it is not the case, $\mathcal{A}'$ completes the rest of the iterations as before. On the other hand, if $K' = K$, it obtains the last round message from $\mathcal{A}$ and forwards $\{\pi_{i,3}\}_{i \notin K'}$ to the challenger.

The reduction obtains $\mathsf{View}_{\mathcal{A}'}$ and $\{\phi_i\}_{i \notin K'}$ from the external challenger. If $\mathcal{A}'$ hadn't completed the execution with the external challenger because $K' \neq K$, then reduction computes the $\mathsf{View}_{\mathcal{A}}$ and the output of the honest party from $\mathsf{View}_{\mathcal{A}'}$. Otherwise, it uses $\{\phi_i\}_{i \notin K'}$ to compute the output of the honest receiver and outputs the view of $\mathcal{A}$ (extracted from $\mathsf{View}_{\mathcal{A}'}$) along with the output of the honest receiver.

We note that if the view of $\mathcal{A}'$ and the output $\{\phi_i\}_{i \notin K'}$ are computed by the external challenger using the real receiver algorithms on input $\{x_i\}_{i \notin K'}$, then the output of the reduction is identically distributed to Hyb_2. Else, it is identically distributed to Hyb_3. Furthermore, the running time of $\mathcal{A}'$ is upper bounded by $\mathsf{poly}(\lambda) \cdot \binom{m}{\lambda} \leq T_2$. Since the changes made to the first iteration are distinguishable with advantage ϵ, it now follows that the reduction breaks the ϵ-simulation security of the inner protocol against malicious senders running in time T_2 and this is a contradiction. $\square$

Lemma 6.3. *Assuming μ-statistical simulation security of the protocol Φ, we have* $\mathsf{Hyb}_3 \approx_{\mu \cdot \lambda \cdot \binom{m}{\lambda}} \mathsf{Hyb}_4$.

Proof. We give the proof that for the first of the $\lambda \cdot \binom{m}{\lambda}$ iterations that the changes made in Hyb_7 are μ-statistically indistinguishable to Hyb_6. The general case of an arbitrary iteration is identical. Assume for the sake of contradiction that these changes made to the first iteration is distinguishable with advantage at least μ. We give a reduction to the security of the protocol Φ.

We interact with the external challenger by giving x as the honest receiver input. We corrupt the sender client and the set of servers indexed by K'. We obtain $\{x_i\}_{i \in K'}$ as the first round message from the receiver client. We use this to generate the first round message of the protocol. If in this iteration $K' = K$, we extract $\{y_i\}_{i \notin K}$ as before. We send $\{y_i\}_{i \notin K}$ as the first round message from the corrupt sender to the honest servers. The external challenger replies with either $f(x, y)$ or abort, and we instruct the honest receiver to output the same. The reduction finally outputs the view of the adversary $\mathcal{A}$ along with the output of the honest receiver.

Note that if the messages of the protocol Φ generated by the external challenger correspond to the real protocol messages, then the output of the reduction is identical to Hyb_3. Else, it is identically distributed to the changes made to the first iteration as described in Hyb_4. It now follows that since these two hybrids are μ-distinguishable, the above reduction breaks the μ-simulation security of the protocol Φ and this is a contradiction. $\square$

Lemma 6.4. *Assuming the t-rewinding, ϵ-security of the extractable commitment scheme* ECom *against receivers running in time T_2, we have that* $\mathsf{Hyb}_7 \approx_{\epsilon \cdot m \cdot \lambda \cdot \binom{m}{\lambda}} \mathsf{Hyb}_6$.

Proof. We consider each rewinding and show that in each of these rewindings, the changes described in Hyb_7 cannot be distinguished from Hyb_6 except with $m \cdot \epsilon$ advantage. We show this for the first rewinding thread and the case for any general rewinding thread is identical.

By a standard averaging argument, we infer that there exists two intermediate hybrids $\mathsf{Hyb}_{6,i}, \mathsf{Hyb}_{6,i-1}$ (described below) which are computationally distinguishable with advantage ϵ. In both hybrids, for every $i' < i$ or if $i' > i$ and $i' \in K'$, we generate $\{\mathsf{Com}_{i',1}, \mathsf{Com}_{i',3}\}$ as a commitment to (x_i, u_i) (where u_i is the random tape used by Sim_S^i). In both hybrids, for every $i' > i$ and $i' \notin K'$, we generate $\{\mathsf{Com}_{i',1}, \mathsf{Com}_{i',3}\}$ as a commitment to a dummy message. The only difference between these two hybrids in the generation of the i-th commitment. Specifically, if $i \notin K'$, we generate $\{\mathsf{Com}_{i.1}, \mathsf{Com}_{i,3}\}$ as a commitment to some dummy message in $\mathsf{Hyb}_{6,i-1}$ and in $\mathsf{Hyb}_{6,i}$, we generate it as a commitment to (x_i, u_i). We now use a distinguisher against $\mathsf{Hyb}_{6,i-1}$ and $\mathsf{Hyb}_{6,i}$ to contradict the computational hiding property of the extractable commitment scheme ECom. Note that if $i \in K'$, then $\mathsf{Hyb}_{6,i}$ and $\mathsf{Hyb}_{6,i-1}$ are identically distributed. Hence, in the rest of the proof, we assume w.l.o.g. that $i \notin K'$.

The reduction generates (x_i, u_i) as described in Hyb_7 and provides this along with a dummy message as the challenge inputs to the external challenger. The reduction obtains $\mathsf{Com}_{i,1}$ from the challenger and generates the rest of the first round protocol messages as described in $\mathsf{Hyb}_{6,i-1}$. Based on the second round message received from the adversary, the reduction checks if $K = K'$ and if it is not the case, it generates the view of $\mathcal{A}$ and the output of the honest party by running the subsequent iterations and outputs them. Otherwise, it continues the interaction with the external challenger. On receiving the second round message from the adversary, the reduction forwards $\mathsf{Com}_{i,2}$ to the challenger and obtains $\mathsf{Com}_{i,3}$. It uses it to generate the final round protocol message as in $\mathsf{Hyb}_{6,i-1}$. To compute $\{y_{i'}\}_{i' \notin K'}$, the reduction rewinds the second round of the protocol $(t-1)$ times by sampling a fresh second round messages from the receiver. On receiving another second round message from $\mathcal{A}$ that includes a possibly different $\widetilde{\mathsf{Com}}_{i,2}$ (in each of the $(t-1)$ rewinds), it again checks if $K = K'$ and if it is not the case, it outputs a special symbol fail. Else, the reduction forwards this to the challenger and obtains $\widetilde{\mathsf{Com}}_{i,3}$. It uses it to compute the final round message of the protocol for each of the $(t-1)$ rewinds. If the adversary outputs a valid third round message in the main thread and in all the $(t-1)$ rewinding threads, then the reduction runs the $\mathsf{Sim}_S^{i'}$ for $\Pi_{i'}$ for each $i' \notin K'$ on these t threads to compute $y_{i'}$. Here, we are using property-2 that to output $y_{i'}$, $\mathsf{Sim}_S^{i'}$ needs t accepting transcripts. It uses this to compute the view of $\mathcal{A}$ and the output of the honest receiver. On the other hand, if the adversary does not output a valid third round message in the main thread, then it outputs the view of $\mathcal{A}$ and the output of the receiver being $\perp$. If the adversary outputs a valid third round message in the main thread, but does not output a valid third round message in one of the rewinding threads, then it outputs a special symbol fail. We now construct a predictor that takes the output of the reduction and does the following: if the output is the special symbol fail then it outputs a random bit. Otherwise, it runs the distinguisher between $\mathsf{Hyb}_{6,i-1}$ and $\mathsf{Hyb}_{6,i}$ on the output of the reduction and outputs whatever the distinguisher outputs.

Let us fix the first round message of the protocol generated using $\mathsf{Com}_{i,1}$. Let E be the event that conditioned on fixing the first round message that $\mathcal{A}$ produces a second and third round message such that $K = K'$ and the third round message is valid. Let p be the probability that E happens. In the case where E happens in both the main thread and all the rewinding threads and in the case where E does not happen in the main thread, the predictor correctly guesses the challenge bit with probability $1/2 + \mu$ (for some non-negligible μ) and this follows from the fact that the distinguisher can distinguish between $\mathsf{Hyb}_{6,i-1}$ and $\mathsf{Hyb}_{6,i}$ with non-negligible advantage. On the other hand, when E happens in the main thread but does not happen in one of the rewinding threads, the probability that the output of the predictor is equal to the challenge bit is $1/2$. Thus, the probability that the output of the predictor is equal to the challenge bit is $(1/2 + \mu)(p^t + (1 - p)) + 1/2(p - p^t)) = 1/2 + \mu(1 - p + p^t) \geq 1/2 + O(\mu)$ (when t is constant). □

6.4.2 Super-Polynomial Simulation Security against Malicious Receivers

Descriptions of SPSim_R. $(\mathsf{SPSim}^1_R, \mathsf{SPExt}_R, \mathsf{SPSim}^2_R)$ are described as follows:

1. For each $i \in [m]$, SPSim^1_R generates the first round message $\pi^S_{i,1}$ using $\mathsf{Sim}^i_{1,R}(1^\lambda)$ and it chooses a set $K \subset [m]$ of size λ uniformly. st consists of the random tape of $\mathsf{Sim}^i_{1,R}(1^\lambda)$ for each $i \in [m]$.
2. SPExt on input $\{\pi^R_{i,1}\}_{i\in[m]}$ does the following:
   (a) For each $i \in [m]$, it runs Ext^i_R on $\pi^R_{i,1}$ to compute x_i.
   (b) It initializes Sim_Φ by corrupting the receiver client and specifies the set of honest servers to be $[m] \setminus K$. It provides $\{x_i\}_{i\in[m]\setminus K}$ as the first round message from the malicious receiver to the honest servers and intercepts the query x that it makes to the ideal functionality. It outputs x and st' to be $\{x_i\}_{i\in[m]}$ and the random tape of Sim_Φ.
3. SPSim^R_2 on input st, st' and $f(x, y)$ and the first two round messages in the protocol:
   (a) It runs Sim_Φ (with the input $\{x_i\}_{i\in[m]\setminus K}$ and random tape from st'). It provides $f(x, y)$ as the output from the trusted functionality. Sim_Φ outputs $\{\phi_i\}_{i\in[m]\setminus K}$.
   (b) It then runs $\mathsf{Sim}^i_{2,R}$ on ϕ_i using the random tape in st and obtains $\pi_{i,3}$ for each $i \in [m]\setminus K$.

Proof of Indistinguishability.

- $\underline{\mathsf{Hyb}_0}$: This hybrid corresponds to $\mathsf{View}_\mathcal{A}(\langle \mathcal{A}(1^\lambda), S(1^\lambda, y)\rangle)$.
- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we make the following changes:
  1. For each $i \in [m]$,
     (a) We generate the first round message $\pi^S_{i,1}$ using $\mathsf{Sim}^i_{1,R}(1^\lambda)$.
     (b) We receive $\pi^R_{i,1}$ from $\mathcal{A}$.
  2. For each $i \notin K$, we run Ext^i_R on $\pi^R_{i,1}$ to compute x_i.
  3. For each $i \notin K$, we generate the last round message $\pi_{i,3}$ using $\mathsf{Sim}^i_{2,R}$ on input ϕ_i (computed using extracted x_i).

  We show in Lemma 6.5 that $\mathsf{Hyb}_0 \approx_\epsilon \mathsf{Hyb}_1$.

- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we make the following changes:
  1. We initialize the simulator Sim_Φ for the protocol Φ by corrupting the receiver client and the set $[m] \setminus K$ to be the set of honest servers.
  2. We obtain $\{x_i\}_{i \notin K}$ as in the previous hybrid and we give this as the first round message from the corrupt client to the honest server. Sim_Φ queries the ideal functionality on input x and we record this value. We provide $f(x, y)$ as the output to Sim_Φ from the ideal functionality.
  3. For each $i \notin K$, Sim_Φ provides ϕ_i as the second round message from the honest servers. We use this to generate the final round message of the protocol.

  We argue in Lemma 6.6 that $\mathsf{Hyb}_1 \approx_\mu \mathsf{Hyb}_2$.

Lemma 6.5. *Assuming the ϵ-super-polynomial simulation security against malicious receivers of the inner protocol, we have* $\mathsf{Hyb}_0 \approx_\epsilon \mathsf{Hyb}_1$.

Proof. Assume for the sake of contradiction that the changes made in the first iteration are distinguishable with advantage greater than ϵ. We now give a reduction that breaks the ϵ-super-polynomial simulation security of inner protocol against malicious receivers.

The reduction computes $(y_1, \ldots, y_m) \leftarrow \Phi_1(y)$ and provides y_i as the challenge input for Π_i. The reduction defines an adversary $\mathcal{A}'$ that internally runs $\mathcal{A}$ and interacts with the external challenger as follows. $\mathcal{A}'$ obtains $\{\pi^S_{i,1}\}_{i \in [m]}$ from the challenger and uses it to compute the first round message in the protocol. It receives the first round message from $\mathcal{A}$ and forwards $\{\pi^R_{i,1}\}_{i \in [m]}$ to the external challenger. It generates the second round message as in the previous hybrid and sends it to the adversary. On receiving the second round message from $\mathcal{A}$, it forwards $\{\pi_{i,2}\}_{i \notin K}$ to the external challenger. It obtains $\{\pi_{i,3}\}_{i \notin K}$ from the challenger and uses it to compute the third round messages from the sender in the protocol.

The reduction obtains $\mathsf{View}_{\mathcal{A}'}$ from the external challenger. The reduction computes the $\mathsf{View}_{\mathcal{A}}$ and outputs it. We note that if the view of $\mathcal{A}'$ is computed by the external challenger using the real sender algorithm on the challenge inputs, then the output of the reduction is identically distributed to Hyb_0. Else, it is identically distributed to Hyb_1. Since the two hybrids are distinguishable with advantage ϵ, it now follows that the reduction breaks the ϵ-super-polynomial simulation security of the inner protocol against malicious receivers running and this is a contradiction. $\square$

Lemma 6.6. *Assuming the μ-statistical simulation security of the protocol Φ, we have* $\mathsf{Hyb}_1 \approx_\mu \mathsf{Hyb}_2$.

Proof. Assume for the sake of contradiction that these two hybrids are distinguishable with advantage at least μ. We give a reduction to the security of the protocol Φ.

We interact with the external challenger by giving y as the honest sender input. We corrupt the receiver client and set $[m] \setminus K$ to be the set of honest servers. We extract $\{x_i\}_{i \notin K}$ as in the previous hybrid and provide it as the first round message from the corrupted receiver client to the honest servers. The challenger replies with $\{\phi_i\}_{i \notin K}$ as the final round message from the honest servers. We use this generate the final round message as before. The reduction finally outputs the view of $\mathcal{A}$ along with the output of the honest sender.

Note that if the messages of the protocol Φ generated by the external challenger correspond to the real protocol messages, then the output of the reduction is identical to Hyb_1. Else, it is identically distributed to Hyb_2. It now follows that since these two hybrids are μ-distinguishable, the above reduction breaks the μ-simulation security of the protocol Φ and this is a contradiction. $\square$

6.4.3 Special Extraction of Malicious Receiver Input

Description of $\mathsf{SPSpecExt}_R$. $\mathsf{SPSpecExt}_R$ does the following:

1. It extracts $\{x'_i\}_{i\in[m]}$ from the extractable commitment using straight-line super-polynomial time extractor.
2. It initializes an empty set C.
3. For each $i \in [m]$, if either (x_i, r_i) is $\perp$ or if $\pi^R_{i,1} \neq \Pi_{i,1}(1^\lambda, R, x_i; r_i)$, then it adds i to C.
4. If $|C| > \lambda$, then it outputs $\perp$.
5. Else, it outputs $\mathsf{Rec}(C \cap ([m] \setminus K), \{x'_i\}_{i\in[m]\setminus K\cup C})$.

We now show that conditioned on the transcript $\mathbb{T}$ being accepting for the sender and the output of SPExt on msg^R_1 is not $\perp$, then $x = x'$. We show this via a hybrid argument.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the output x of SPExt.
- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we make the following changes:
  - For each $i \in [m]$, we run the straight-line extractor on the commitments to compute the value (x_i, r_i) committed in the extractable commitment Com^i_1.
  - Initialize an empty set C.
  - For each $i \in [m]$, if either (x_i, r_i) is $\perp$ or if $\pi^R_{i,1} \neq \Pi_{i,1}(1^\lambda, R, x_i; r_i)$, then add i to C.
  - If $|C| > \lambda$, then output $\perp$ instead of x.

  Note that since K is uniformly chosen, it follows from a similar argument to Claim 4.4 that the sender outputs reject at the end of the protocol with overwhelming probability and this is a contradiction to the fact that the sender is accepting.
- $\underline{\mathsf{Hyb}_2}$: In this hybrid, if $|C| \leq \lambda$, we output $x' = \mathsf{Rec}(C \cap ([m] \setminus K), \{x'_i\}_{i\in[m]\setminus K\cup C})$ instead of x if $x \neq \perp$. It follows from the correctness of the protocol Π_i that for every $i \notin C$, we have that x'_i is same as x_i. Note that if $x \neq \perp$ and since $|C| \leq \lambda$, then by the property-3 of the outer protocol, x' is same as x. Note that x' output in Hyb_2 is identical to the output of $\mathsf{SPSpecExt}$.

6.5 Existence of k accepting Transcripts Extractor.

The description of Ext_R is exactly same as $\mathsf{SPSpecExt}$ except that it uses the k-accepting transcripts to extract $\{(x_i, r_i)\}_{i\in[m]}$ in polynomial time rather than super-polynomial time by relying the k-rewinding extractor of ECom. It now follows from the property of ECom that the values extracted by Ext_R and $\mathsf{SPSpecExt}$ are identical (except with negligible probability) and thus, the output of both is the same except with negligible probability.

6.6 Additional Properties

For constructing a watchlist protocol, we require that the 3-round protocol with special extraction to satisfy an additional property described below.

2-Rewinding Sender Security against Sub-Exponential Adversaries. We require that this protocol to be secure against any malicious receiver that could rewind an honest sender twice by giving possibly different second round messages in each rewind. We note that the same construction and the proof of security can be adapted to give such a protocol based on any 2-rewinding sender secure inner protocol. As mentioned in Section 5.5, such an inner protocol can be constructed by relying on a 3-round OT protocol that satisfies 2-rewinding sender security. In section 4.5, we also described how to construct such an OT protocol. We need the 2-rewinding sender security to hold even against adversaries that run in time which is polynomial in the running time of SPExt_R. This just follows from assuming that the 2-rewinding sender secure inner protocol is secure against sub-exponential receivers which in turn follows from sub-exponential hardness of the underlying 2-message OT protocol.

Delayed Function Selection. As in the case of the 2PC protocol described in Section 5, we require this protocol to also have delayed function selection. This property is directly inherited from the underlying inner protocol. Further, if we require both delayed function selection and 2-rewinding sender security then we require the outer protocol's first round message to be reusable 3 times. In Section 5.5, we described how to modify the protocol from [IKP10, Pas12] to satisfy this property.

Existence of Straight-Line SPS extractor against Malicious Senders. This property is directly inherited from the underlying inner protocol.

7 The Watchlist Protocol

In this section, we formally construct and prove security of three-round watchlist protocol. Recall that in the watchlist protocol, each ordered pair of parties P_i and P_j invoke a ℓ-out-of-m OT functionality where P_i acts as the receiver and P_j acts as the sender. Specifically, the private input of party P_j in this OT instance consists of of $\mathbf{x}_j$ which is the vector of sender inputs of dimension m and the private input of party P_i is K_i which is a subset of $[m]$ of size ℓ. The output to party P_i consists of $\{\mathbf{x}_{j,k}\}_{k \in K_i}$. We note that in the watchlist protocol, every honest party P_i uses the same K_i in each instance of the OT functionality when acting as the receiver and same $\mathbf{x}_i$ in each instance when acting as the sender. However, the corrupted party P_i may choose different K_i and $\mathbf{x}_i$ for each instance when acting as the receiver and the sender respectively. For ease of notation, whenever we use K_i as the receiver input of a corrupted party P_i, we actually mean a set of subsets $\{K_{i,j}\}_{j \in H}$. Similarly, whenever we use $\mathbf{x}_j$ as the sender input of a corrupted party P_j, we actually mean set of vectors, one for each honest party.

7.1 Definitions

Before we proceed to the formal definition of the watchlist protocol, we give an informal overview of the various properties that the protocol needs to satisfy.

1. The first requirement is the existence of a straight-line super-polynomial time simulator $\mathsf{Sim}_{\mathsf{WL}}$ that has oracle access to the watchlist functionality and produces a view of the adversary that is computationally indistinguishable to the real world. This requirement is same as standard SPS security. Here, it is crucial that the simulator is straight-line i.e., it does not rewind the adversary.

2. The second property is about the existence of an "alternate" extraction mechanism of the malicious receiver inputs. Specifically, we require that if the output of all the honest parties when acting as the sender is not $\perp$ in the protocol, then there exists an alternate super-polynomial time extractor $\mathsf{SPExt}_{\mathsf{WL},R}$ that extracts the adversarial receiver inputs using the accepting transcript. Further, for each corrupted party, these inputs are the same as the ones extracted by $\mathsf{Sim}_{\mathsf{WL}}$ except in the case that it is $\perp$.

3. The third property is about the existence of polynomial-time rewinding extractor (that rewinds the adversary until it obtains k accepting transcripts) and outputs the malicious receiver inputs that is identical to the one output by $\mathsf{SPExt}_{\mathsf{WL},R}$. For technical reasons, we need to separate out the existence of a polynomial time rewinding extractor and super-polynomial time extractor in the alternate extraction mechanism.

4. The fourth property is about the one-rewinding sender non-malleability. Roughly speaking, it requires that adversarial sender inputs cannot depend on the honest party sender inputs even if the adversary is allowed to rewind the second and third round message of the protocol once.

Definition 7.1 (Extractable (n, m, ℓ)-Watchlists). *Fix any polynomials $n = n(\lambda), m = m(\lambda), \ell = \ell(\lambda)$. An extractable (n, m, ℓ)-watchlist is a protocol that achieves the simultaneous n-party m-choose-ℓ OT functionality with the following security guarantees:*

1. ***Real-Ideal Security with Straight-line SPS simulator.*** *There exists a (stateful) straight-line super-polynomial time simulator $\mathsf{Sim}_{\mathsf{WL}}$ such that for any (stateful) adversary $\mathcal{A}$ that is corrupting an arbitrary subset M of the parties and for any choice of honest party inputs $\{\mathbf{x}_j, K_j\}_{j \in H}$ (where H denotes the set of honest parties, $\mathbf{x}_j$'s denote the sender inputs of party j, and K_j's denote the set of executions that player j watches), we have the following two distributions are computationally indistinguishable:*

   (a) View of the adversary and the output of all the honest parties H in the real execution of the protocol.

   (b) $\mathsf{Ideal}_{SPS}(1^\lambda, M, \mathcal{A}, \mathsf{Sim}_{\mathsf{WL}})$ where Ideal_{SPS} is given in Figure 6.

   Furthermore, the distribution of the messages generated by $\mathsf{Sim}_{\mathsf{WL}}$ on behalf of honest receivers is identically distributed to the real receiver messages with dummy inputs.

2. ***Special Extraction of the Malicious Receiver Input.*** *There exists a super-polynomial time extractor $\mathsf{SPExt}_{\mathsf{WL},R}$ such that for any adversary $\mathcal{A}$ corrupting a subset M of the parties and for any choice of honest party inputs $\{\mathbf{x}_j, K_j\}_{j \in H}$, the probability that the following experiment outputs 1 is negligible:*

   (a) Sample a transcript $\mathbb{T}$ from Ideal_{SPS} experiment and let $\{\sigma_j\}_{j \in H}$ be the output of the honest parties.

1. Run $\mathsf{Sim}_{\mathsf{WL}}(1^\lambda, M)$ to obtain $\{\mathsf{msg}_1^j\}_{j\in H}$ and send this to $\mathcal{A}$. Receive $\{\mathsf{msg}_1^i\}_{i\in M}$ from $\mathcal{A}$.
2. Run $\mathsf{Sim}_{\mathsf{WL}}(\{\mathsf{msg}_1^i\}_{i\in M})$ to obtain $(\{K_i\}_{i\in M}, \mathsf{st})$.
3. Compute the output of the watchlist received by the parties in M when the honest sender inputs are $\{\mathbf{x}_j\}_{j\in H}$ and the malicious receiver inputs are $\{K_i\}_{i\in M}$. Let $\{\sigma_i\}_{i\in M}$ be this output.
4. For each $r \in \{2,3\}$:
   (a) Run $\mathsf{Sim}_{\mathsf{WL}}(\{\sigma_i\}_{i\in M}, \mathsf{st}, \{\mathsf{msg}_k^i\}_{k\in[r-1], i\in M})$ to obtain $\{\mathsf{msg}_r^j\}_{j\in H}$ and send this to $\mathcal{A}$. Receive $\{\mathsf{msg}_r^i\}_{i\in M}$ from $\mathcal{A}$.
5. Run $\mathsf{Sim}_{\mathsf{WL}}(\{\mathsf{msg}_k^i\}_{i\in M, k\in[3]})$ to obtain $\{\mathbf{x}_i\}_{i\in M}$.
6. Compute the output of the watchlist received by the parties in H when the honest receiver inputs are $\{K_j\}_{j\in H}$ and the malicious sender inputs are $\{\mathbf{x}_i\}_{i\in M}$. Let $\{\sigma_j\}_{j\in H}$ be this output.
7. Output view of $\mathcal{A}$ and $\{\sigma_j\}_{j\in H}$.

Figure 6: Description of Ideal_{SPS}.

(b) If $\sigma_j = \perp$ for each $j \in H$ in Ideal_{SPS} experiment, then output of the experiment is 0.

(c) Else, run $\mathsf{Sim}_{\mathsf{WL}}(\{\mathsf{msg}_1^i\}_{i\in M})$ (where $\{\mathsf{msg}_1^i\}_{i\in M} \in \mathbb{T}$) to obtain $(\{K_i\}_{i\in M}, \mathsf{st})$.

(d) Run $\mathsf{SPExt}_{\mathsf{WL},R}(\mathbb{T})$ to obtain $\{K_i'\}_{i\in M}$.

(e) The output of the experiment is 1 if and only if there exists an $i \in H$ such that $K_i' \neq K_i$ whenever $K_i \neq \perp$.

3. ***Existence of k accepting Transcript Extractor.*** *There exists a polynomial time machine $\mathsf{Ext}_{\mathsf{WL},R}$ such that on input any k transcripts $\mathbb{T}_1, \ldots, \mathbb{T}_k$ with common first message such that in each of the transcript the output of the honest parties is not $\perp$ outputs $\{\overline{K}_j\}_{j\in H}$ such that $\{\overline{K}_j\}_{j\in H} = \mathsf{SPExt}_{\mathsf{WL},R}(\mathbb{T}_1)$ with overwhelming probability.*

4. ***One-Rewinding Sender Non-Malleability.*** *We require the existence of an (expected) PPT algorithm $\mathsf{Ext}_{\mathsf{WL},S}$ such that for any 1-rewinding adversary $\mathcal{A}$ corrupting any set M of the parties (by 1-rewinding, we refer to an adversary that is allowed to rewind the second and third round message of the protocol once) and for any choice of honest party inputs $\{\mathbf{x}_j, K_j\}_{j\in H}$ such that the following two distributions are computationally indistinguishable against adversaries that run in time which is polynomial in the running time of $\mathsf{SPSim}_{\mathsf{WL},R}$:*

   (a) Consider the Ideal_{SPS} experiment in Figure 6 with the 1-rewinding adversary $\mathcal{A}$ (i.e., step-4 in the experiment is repeated once more). Let us denote the first execution with the adversary as the main thread and the rewinding execution with $\mathcal{A}$ as the rewind thread. After step-4, run $\mathsf{Sim}_{\mathsf{WL}}$ on the messages generated in the main thread to compute $\{\mathbf{x}_i\}_{i\in M}$. Output the view of the adversary $\mathcal{A}$ and $\{\mathbf{x}_i\}_{i\in M}$.

   (b) Sample uniform random tape $\{r_j\}_{j\in H}$ and execute the protocol honestly with the 1-rewinding adversary $\mathcal{A}$ using the honest inputs $\{K_j, \mathbf{x}_j\}_{j\in H}$ with the above random tape. Run $\mathsf{Ext}_{\mathsf{WL},S}(1^\lambda, \{K_j, \mathbf{x}_j, r_j\}_{j\in H})$ to obtain $\{\mathbf{x}_i\}_{i\in M}$. Output view of the adversary in the above honest execution along with $\{\mathbf{x}_i\}_{i\in M}$.

7.2 Construction

Our construction is described in Figure 7, and makes use of the following ingredients:

- A 3 round two-party secure computation protocol Π satisfying Definition 6.1 with delayed-function selection for NC^1 circuits and 2-rewinding sender security.
- An information-theoretic $m(\lambda) \cdot \ell(\lambda)$ non-malleable coding scheme satisfying Definition 3.7.
- A low-depth proof for P according to Definition 3.9.
- 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 7. 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 according to Definition 7.1.

Theorem 7.2. *Let λ denote the security parameter, and $m = m(\lambda), k = k(\lambda), \ell = \ell(\lambda)$ be arbitrary polynomials. There exists a 3 round ℓ non-malleable $\binom{m}{k}$ oblivious transfer protocol satisfying Definition 7.1 that makes black-box use of any 3 round two-party secure computation protocol Π satisfying Definition 6.1 with 2-rewinding sender security, and any existentially unforgeable signature scheme.*

Proof of Theorem 7.2. We observe that properties 2 and 3 carry over from the properties of the underlying two-party computation protocol, and 1 is implied by 4 together with security of the two-party protocol against malicious adversaries (following [IKSS21]). Thus, our key goal is to prove that the protocol satisfies property 4. To keep exposition simple, we prove this property against polynomial time distinguishers. We note that indistinguishability against distinguishers running in time which is polynomial in the running time of $\mathsf{SPExt}_{\mathsf{WL},R}$ follows directly from the 2-rewinding sender security of the underlying 2PC protocol against such distinguishers.

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. Towards proving that our protocol satisfies property 1, we will prove that there exists a PPT algorithm $\mathsf{Sim\text{-}Ext}$, that with black-box access to the MIM, and to ℓ copies of the ideal OT functionality $\mathbf{OT} = \{\mathsf{OT}_j(\{\mathsf{m}_{i,j}\}_{i\in[m]}, \cdot)\}_{j\in[\ell]}$ and with input $\{K_j\}_{j\in[\ell]}$, simulates an execution of the protocol with the MIM and extracts all the inputs $\{(\{\widetilde{\mathsf{m}}_{i,j}\}_{i\in[m]})\}_{j\in[\ell]}$ used by the MIM in the executions where the MIM is sender. We will prove that the 1-rewinding view output by $\mathsf{Sim\text{-}Ext}$, that we denote by $\mathsf{Ideal}_{\mathsf{MIM}}(\{\mathsf{m}_{i,j}\}_{i\in[m],j\in[\ell]}, \{K_j\}_{j\in[\ell]})$ will be such that

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

where the expression on the left denotes the joint distribution of the view and messages committed by a 1-rewinding adversary in an interaction where honest senders $\mathcal{S}_j$ have inputs $\{\mathsf{m}_{i,j}\}_{i\in[m]}$, and honest receivers $\mathcal{R}_j$ have inputs K_j.

To prove indistinguishability, we define a sequence of hybrid experiments, where the first one outputs the distribution $\mathsf{Real}_{\mathsf{MIM}}\langle\{\mathcal{S}_j(\{\mathsf{m}_{i,j}\}_{i\in[m]})\}_{j\in[\ell]}, \{\mathcal{R}_j(K_j)\}_{j\in[\ell]}$ and the final one outputs the distribution $\mathsf{Ideal}_{\mathsf{MIM}}(\{\mathsf{m}_{i,j}\}_{i\in[m],j\in[\ell]}, \{K_j\}_{j\in[\ell]})$. Formally, these hybrids are defined as follows:

Inputs: Sender $\mathcal{S}$ has inputs $\{\mathsf{m}_j\}_{j\in m}$ 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
     $$(\mathsf{L}_{i,j}, \mathsf{M}_{i,j}, \mathsf{R}_{i,j}) = \mathsf{NM.Code}((vk|\mathsf{m}_j); r_{i,j}).$$
   - Set instance $x = (vk, \{(\mathsf{L}_{i,j}, \mathsf{M}_{i,j}, \mathsf{R}_{i,j}, \mathsf{m}_j)\}_{i\in[\lambda],j\in[m]})$ and language
     $$\mathcal{L} = \big\{(vk, \{(\mathsf{L}_{i,j}, \mathsf{M}_{i,j}, \mathsf{R}_{i,j}, \mathsf{m}_j)\}_{i\in[\lambda],j\in[m]}) : \\ \forall i \in [\lambda], j \in [m], \mathsf{NM.Decode}(\mathsf{L}_{i,j}, \mathsf{M}_{i,j}, \mathsf{R}_{i,j}) = (vk|\mathsf{m}_j)\big\}.$$

   Compute $\mathsf{ldp} = \mathsf{LDP.Prove}(x, \mathcal{L})$.
2. For each $i \in [\lambda]$, $\mathcal{R}$ picks $\mathsf{c}_i \leftarrow \{0, 1, 2\}$.
3. 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 $(\mathsf{c}_1, \ldots, \mathsf{c}_\lambda)$ chosen in round 2.
   - $\mathcal{S}$ plays the sender with input (x, ldp), where x is parsed as $(vk, \{\mathsf{m}_j, (\mathsf{L}_{i,j}, \mathsf{M}_{i,j}, \mathsf{R}_{i,j})\}_{i\in[\lambda],j\in[m]}$.
   - The functionality $\mathcal{F}$ on input $(vk, \{\mathsf{m}_j, \mathsf{L}_{i,j}, \mathsf{M}_{i,j}, \mathsf{R}_{i,j}\}_{i\in[\lambda],j\in[m]}, K, \{\mathsf{c}_i\}_{i\in[\lambda]})$ generates an output as follows:
     - If $\mathsf{LDP.Verify}(x, \mathsf{ldp}) \neq 1$, output $\perp$. Otherwise set $\mathsf{out} = vk, \{\mathsf{m}_j\}_{j\in K}$.
     - Additionally, for every $i \in [\lambda]$, if $c_i = 0$, append $(\{\mathsf{L}_{i,j}\}_{j\in[m]})$ to out, if $c_i = 1$, append $(\{\mathsf{M}_{i,j}\}_{j\in[m]})$ 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_1, Π_3). It sets $\sigma_1 = \mathsf{Signature.Sign}(\Pi_1, \mathsf{id}_S, sk)$, $\sigma_3 = \mathsf{Signature.Sign}(\Pi_3, \mathsf{id}_S, sk)$ where id_S is the identity of the sender. It sends (σ_1, σ_3) to $\mathcal{R}$.
4. $\mathcal{R}$ obtains output out and parses $\mathsf{out} = (vk, \{\mathsf{m}_j\}_{j\in K}, \cdot)$. It outputs $\{\mathsf{m}_j\}_{j\in K}$ iff $\mathsf{Signature.Verify}(\sigma_1, \Pi_1, \mathsf{id}_S, vk) \wedge \mathsf{Signature.Verify}(\sigma_3, \Pi_3, \mathsf{id}_S, vk) = 1$, otherwise outputs $\perp$.

Figure 7: $\ell(\lambda)$ Non-Malleable $m(\lambda)$-choose-$k(\lambda)$ Oblivious Transfer

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 $\{\mathsf{m}_{i,j}\}_{i\in[m]}$ respectively and ℓ honest receivers on the right with inputs $(\{K_j\}_{j\in[\ell]})$ respectively. The output of this hybrid is $\mathsf{Real}_{\mathsf{MIM}}\langle\{\mathcal{S}_j(\{\mathsf{m}_{i,j}\}_{i\in[m]})\}_{j\in[\ell]}, \{\mathcal{R}_j(K_j)\}_{j\in[\ell]}$.

Hyb_1 : This experiment modifies Hyb_1 by introducing an additional abort condition. Specifically, the experiment first executes the complete protocol corresponding to the real execution of the MIM exactly as in Hyb_0 (including rewinding the MIM once) to obtain the distribution $\mathsf{Real}_{\mathsf{MIM}}\langle\{\mathcal{S}_j(\{\mathsf{m}_{i,j}\}_{i\in[m]})\}_{j\in[\ell]},\{\mathcal{R}_j(K_j)\}_{j\in[\ell]}\rangle$.

Let $p(\lambda)$ denote the probability that the MIM completes this execution without aborting. Set $\gamma(\lambda) = \max\big(\lambda, p^{-2}(\lambda)\big)$. With the first two rounds of the transcript fixed, the rewind the right execution up to $\gamma^2(\lambda)$ times, picking inputs $(\mathsf{c}^j_1,\ldots,\mathsf{c}^j_\lambda)$ for each of the ℓ receivers $\{\mathcal{R}_j\}_{j\in[\ell]}$ independently and uniformly at random in every run. If there exist two rewinding threads where the MIM completes the protocol execution, denote the inputs chosen by the challenger on behalf of the honest receiver in these rewinding threads by $(\mathsf{c}'^j_1,\ldots,\mathsf{c}'^j_\lambda)$ and $(\mathsf{c}''^j_1,\ldots,\mathsf{c}''^j_\lambda)$ respectively. For every $j\in[\ell]$, let index $\alpha_j\in[\lambda]$ be such that $\mathsf{c}^j_{\alpha_j}=0,\mathsf{c}'^j_{\alpha_j}=1,\mathsf{c}''^j_{\alpha_j}=2$.

Additionally for every $j\in[\ell], i\in[m]$, use $(\widetilde{\mathsf{L}}^j_{\alpha_j,i},\widetilde{\mathsf{M}}^j_{\alpha_j,i},\widetilde{\mathsf{R}}^j_{\alpha_j,i})$ obtained as output from the main and rewinding executions respectively to compute $\widetilde{\mathsf{m}}^j_i = \mathsf{NM.Decode}(\widetilde{\mathsf{L}}^j_{\alpha_j,i},\widetilde{\mathsf{M}}^j_{\alpha_j,i},\widetilde{\mathsf{R}}^j_{\alpha_j,i})$.

If no such rewinding thread exists, or if there exists $j\in[\ell]$ for which there does not exist $\alpha\in[\lambda]$ such that $\mathsf{c}^j_\alpha=0,\mathsf{c}'^j_\alpha=1,\mathsf{c}''^j_\alpha=2$, then set $\widetilde{\mathsf{m}}^j_i=\bot$ for all $i\in[m]$. Now, the output of this hybrid is the joint distribution

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

Lemma 7.3. *For every unbounded distinguisher $\mathcal{D}$ and large enough $\lambda\in\mathbb{N}$,*

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

Proof. Since the MIM's inputs $\{\widetilde{\mathsf{m}}^j_i\}_{j\in[\ell]}$ are committed in round 1 of the protocol, then conditioned on the adversary providing a non-aborting transcript in rewinding executions in Hyb_1, by simulation security of the $\mathsf{2pc}$, $\{(\widetilde{\mathsf{m}}^j_i\}_{j\in[\ell]}$ are correctly extracted.

Therefore, to prove this lemma it suffices to show that two rewinding executions (with a non-aborting transcript) can be found within $\gamma^2(\lambda)$ 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 $\gamma^2(\lambda)-1$ out of the $\gamma^2(\lambda)$ trials abort is $\mathsf{negl}(\lambda)$. $\square$

Hyb_2: This experiment modifies Hyb_1 to execute the superpolynomial simulator of Π in all sessions where the MIM is a receiver. Specifically, in these executions, instead of the honest sender strategy with input $\{\mathsf{m}^j_i\}_{i\in[m],j\in[\ell]}$, we execute the superpolynomial simulator $\mathsf{Sim\text{-}2PC}_{\mathsf{Sen}}^{\mathsf{MIM},\mathcal{F}(\mathsf{inp}_{\mathcal{S}^j},\cdot)}$ where

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

$\mathsf{Sim\text{-}2PC}_{\mathsf{Sen}}$ expects round 1 and round 2 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 round of the transcript fixed, the challenger rewinds the right execution up to $\gamma^2(\lambda)$ times, picking inputs $(\mathsf{c}^j_1, \ldots, \mathsf{c}^j_\lambda)$ for $\mathcal{R}^j$ independently and uniformly at random in every run, and generating messages in the left execution by running the simulator $\mathsf{Sim\text{-}2PC}_{\mathsf{Sen}}$ each time.

If there exist two rewinding executions where the MIM completes the protocol, denote the inputs chosen by the challenger on behalf of the honest receiver in this rewinding thread by $(\mathsf{c}'^j_1, \ldots, \mathsf{c}'^j_\lambda)$ and $(\mathsf{c}''^j_1, \ldots, \mathsf{c}''^j_\lambda)$ respectively. For every $j \in [\ell]$, let index $\alpha_j \in [\lambda]$ be such that $\mathsf{c}^j_{\alpha_j} = 0, \mathsf{c}'^j_{\alpha_j} = 1, \mathsf{c}''^j_{\alpha_j} = 2$. Additionally for every $j \in [\ell], i \in [m]$, use $(\widetilde{\mathsf{L}}^j_{\alpha_j,i}, \widetilde{\mathsf{M}}^j_{\alpha_j,i}, \widetilde{\mathsf{R}}^j_{\alpha_j,i})$ obtained as output from the main and the two rewinding executions respectively to compute $\widetilde{\mathsf{m}}^j_i = \mathsf{NM.Decode}(\widetilde{\mathsf{L}}^j_{\alpha_j,i}, \widetilde{\mathsf{M}}^j_{\alpha_j,i}, \widetilde{\mathsf{R}}^j_{\alpha_j,i})$. If no such rewinding thread exists, or if there exists $j \in [\ell]$ for which there does not exist $\alpha \in [\lambda]$ such that $\mathsf{c}^j_\alpha = 0, \mathsf{c}'^j_\alpha = 1, \mathsf{c}''^j_\alpha = 2$, then abort. The output of this hybrid is the joint distribution:

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

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

Lemma 7.4. *Assuming 2-rewinding secure two party computation according to Definition 6.1, 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. 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 $\mathsf{Sim\text{-}2PC}_{\mathsf{Sen}}^{\mathsf{MIM},\mathcal{F}(\mathsf{inp}_{\mathcal{S}^j},\cdot)}$ where

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

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 $\lambda \in \mathbb{N}$,

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

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 2-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.
  - Obtain the first round sender message from the $\mathsf{2pc}$ challenger, and forward this to the MIM as $\mathcal{S}^{j^*}$'s message in the j^{*th} left execution. In addition, generate the first round messages according to receiver strategy with inputs $\{K^j\}_{j\in[\ell]}$ for the right execution. Obtain the first round message from the MIM, which includes a (malicious) sender message for the right execution and a (malicious) receiver message for the left execution. Output the MIM's receiver message in the j^{*th} left execution to the challenger of the $\mathsf{2pc}$.
  - Generate the second round message for the right execution according to honest receiver strategy, and obtain the second round message for the left execution from the challenger. Forward the MIM's message in left session j^* to the challenger.

  - Obtain the third 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. Generate messages for the right executions using honest receiver strategy. Obtain the third round message from the MIM for the right execution.

- Next, $\mathcal{A}$ rewinds $\mathcal{R}'$ twice with fixed first round, and obtains MIM outputs as follows.
  - Run the second round with honest receiver strategy on the right, and obtain challenger messages on the left. Obtain the second round message from the MIM, and output the MIM's message in session j^* to the challenger.
  - Obtain the third 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 third round messages from the MIM.

- If none of the executions abort, for every $j \in [\ell]$, find $\alpha_j \in [\lambda]$ such that $\mathsf{c}^j_{\alpha_j} = 0, \mathsf{c}'^j_{\alpha_j} = 1, \mathsf{c}''^j_{\alpha_j} = 2$. If these do not exist, abort. Otherwise use the outputs of the two-party computation protocol to compute $\widetilde{\mathsf{m}}^j_i = \mathsf{NM.Decode}(\widetilde{\mathsf{L}}^j_{\alpha_j,i}, \widetilde{\mathsf{M}}^j_{\alpha_j,i}, \widetilde{\mathsf{R}}^j_{\alpha_j,i})$ for $i \in [m], j \in [\ell]$. Else, set $\widetilde{\mathsf{m}}^j_i = \bot$ for $i \in [m], j \in [\ell]$

- $\mathcal{A}$ outputs the entire view of $\mathcal{R}'$ together with $\{\widetilde{\mathsf{m}}^j_i\}_{i\in[m],j\in[\ell]}$. 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, 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_2.

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 $\lambda \in \mathbb{N}$:

$$\Big|\Pr[\mathcal{D} = 1 \ \wedge \ \neg\mathsf{abort}|\mathsf{Hyb}_1] - \Pr[\mathcal{D} = 1 \ \wedge \ \neg\mathsf{abort}|\mathsf{Hyb}_2]\Big| \geq \frac{1}{p^2(\lambda) \cdot q(\lambda)}$$

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 $\lambda \in \mathbb{N}$:

$$\Big|\Pr[\mathcal{D}(\mathsf{Dist}_1) = 1] - \Pr[\mathcal{D}(\mathsf{Dist}_2) = 1]\Big| \geq \frac{1}{p^2(\lambda) \cdot q(\lambda)},$$

where Dist_1 and Dist_2 denote the real and ideal distributions of the underlying 2-party computation protocol under 2-rewinding security. This implies that $\mathcal{D}$ contradicts 2-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 $\perp$. By existential unforgeability of the signature scheme, given any PPT adversary MIM, Hyb_2 and Hyb_3 are computationally 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 $(\mathsf{L}^j_{\alpha,i}, \mathsf{M}^j_{\alpha,i}, \mathsf{R}^j_{\alpha,i}) \leftarrow \mathsf{NM.Sim}(1^{p(\lambda)})$, and set

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

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 $\{\mathsf{OT}^j(\mathsf{m}^j_i, \mathsf{m}^j_i, \cdot)\}_{j\in[\ell]}$. Moreover, this hybrid runs the super-polynomial simulator of the two-party computation protocol, which can be split into a straight-line simulator that extracts adversarial receiver input from the first round, and then a rewinding-based expected polynomial-time simulator that extracts adversarial sender input. The latter can also be replaced by a straight-line superpolynomial simulator that extracts the adversarial sender-input by running the straight-line superpolynomial simulator of the two-party computation protocol. Finally, as long as the underlying two-party computation protocol has its ideal distribution be identical to an honest execution with dummy inputs, the same is true for our protocol. Therefore, the output of this hybrid is identical to the ideal distribution $\mathsf{Ideal}_{\mathsf{MIM}}(\{\mathsf{m}^j_i\}_{i\in[m],j\in[\ell]}, \{K^j\}_{j\in[\ell]})$.

Lemma 7.5. *Assuming $m(\lambda) \cdot \ell(\lambda)$ symmetric non-malleable codes satisfying Definition 3.7, 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 $(\mathsf{L}^j_{k,i}, \mathsf{M}^j_{k,i}, \mathsf{R}^j_{k,i}) \leftarrow \mathsf{NM.Code}(0)$.

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 set of tampering functions $(f_{\mathsf{MIM}}, g_{\mathsf{MIM}}, h_{\mathsf{MIM}})$, and a set of additional functions $(w_{\mathsf{MIM}}, y_{\mathsf{MIM}}, z_{\mathsf{MIM}})$. Before defining them, we define a shared state for these functions, 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 2.
- 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 $\{\widetilde{K}^j\}_{j\in[\ell]}$ that correspond to the receiver's inputs in the left execution.
- Rewind the second round twice with uniformly and independently chosen $\{\mathsf{c}'^j_1, \ldots, \mathsf{c}'^j_\lambda\}_{j\in[\ell]}$ and $\{\mathsf{c}''^j_1, \ldots, \mathsf{c}''^j_\lambda\}_{j\in[\ell]}$ respectively in each rewind. If for every $j \in [\ell(\lambda)]$, there exists $\alpha_j \in [\lambda]$ such that $\mathsf{c}^j_{\alpha_j} = 0, \mathsf{c}'^j_{\alpha_j} = 1, \mathsf{c}''^j_{\alpha_j} = 2$, continue, otherwise abort.
- Obtain the rewinding message of the adversary in the second round (with the same first round prefix), as well as $(\bar{\mathsf{c}}_1, \ldots, \bar{\mathsf{c}}_n)$ and $(\widehat{\mathsf{c}}_1, \ldots, \widehat{\mathsf{c}}_n)$ that correspond to the receiver's chosen functions in the j^*th left session in this rewinding execution.
- If $\widetilde{\mathsf{c}}_{k^*}, \bar{\mathsf{c}}_{k^*}$ and $\widehat{\mathsf{c}}_{k^*}$ are all different, continue. Otherwise, abort.
- Generate $(\mathsf{L}^j_{k,i}, \mathsf{M}^j_{k,i}, \mathsf{R}^j_{k,i})$ for every $(i, j, k) \in [m]\times[\ell]\times[\lambda]\setminus\{i^*, j^*, k^*\}$ 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 2 in the main the rewinding threads, and also output (i^*, j^*, k^*), and the values $(\mathsf{L}^j_{k,i}, \mathsf{M}^j_{k,i}, \mathsf{R}^j_{k,i})_{(i,j,k)\in[m]\times[\ell]\times[\lambda]\setminus\{i^*,j^*,k^*\}}$.
- Additionally, output the receiver's inputs $\{\widetilde{K}^j, \widetilde{\mathsf{c}}^j_1, \ldots, \widetilde{\mathsf{c}}^j_\lambda\}_{j\in[\ell]}$ and also output the sender's inputs $\{sk^j, vk^j, \{\mathsf{m}^j_i\}_{i\in[m]}\}_{j\in[\ell]}$, along with randomness r.

The functions $f_{\mathsf{MIM},i,j}, g_{\mathsf{MIM},i,j}$ and $h_{\mathsf{MIM},i,j}$ correspond to tampering functions, and are defined as follows.

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

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

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

  It then invokes $\mathsf{Sim\text{-}2PC}_{\mathsf{Sen}}$ using randomness r on out to generate the third round message of the protocol transcript in the thread corresponding to the receiver challenge being 0. It outputs the value $\mathsf{L}^j_{\alpha_j,i}$ or $\mathsf{M}^j_{\alpha_j,i}$ or $\mathsf{R}^j_{\alpha_j,i}$ obtained from the MIM.

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

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

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

  It then invokes $\mathsf{Sim\text{-}2PC}_{\mathsf{Sen}}$ using randomness r on out to generate the third round message of the protocol transcript in the thread corresponding to the receiver challenge being 1. It outputs the value $\mathsf{L}^j_{\alpha_j,i}$ or $\mathsf{M}^j_{\alpha_j,i}$ or $\mathsf{R}^j_{\alpha_j,i}$ obtained from the MIM.

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

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

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

  It then invokes $\mathsf{Sim\text{-}2PC_{Sen}}$ using randomness r on out to generate the third round message of the protocol transcript in the thread corresponding to the receiver challenge being 2. It outputs the value $\mathsf{L}^j_{\alpha_j,i}$ or $\mathsf{M}^j_{\alpha_j,i}$ or $\mathsf{R}^j_{\alpha_j,i}$ obtained from the MIM.

The functions $w_{\mathsf{MIM}}, y_{\mathsf{MIM}}, z_{\mathsf{MIM}}$ generate the threads themselves and are defined as follows.

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

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

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

  It then invokes $\mathsf{Sim\text{-}2PC_{Sen}}$ on out to generate the third round message of the protocol transcript in the thread corresponding to receiver left challenge being 0. It outputs the resulting transcript as one thread in the view of the MIM.

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

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

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

  It then invokes $\mathsf{Sim\text{-}2PC_{Sen}}$ on out to generate the third round message of the protocol transcript in the thread corresponding to receiver left challenge being 1. It outputs the resulting transcript as another thread in the view of the MIM.

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

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

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

  It then invokes $\mathsf{Sim\text{-}2PC_{Sen}}$ on out to generate the third round message of the protocol transcript in the thread corresponding to receiver left challenge being 2. It outputs the resulting transcript as another thread in the view of the MIM.

Note that there is a fixed set of permutations $\sigma_{i,j}$ such that $f_{\mathsf{MIM},i,j}, g_{\mathsf{MIM},i,j}, h_{\mathsf{MIM},i,j}$ can be relabeled as functions $F_{i,j}, G_{i,j}, H_{i,j}$ such that $F_{i,j}$ outputs $\widetilde{L}$ values, $G_{i,j}$ outputs $\widetilde{M}$ values, and $H_{i,j}$ outputs $\widetilde{R}$ values.

By Definition 3.7 of ℓ augmented non-malleable codes, we have that for every permutation σ and σ' on $\mathsf{L}, \mathsf{M}, \mathsf{R}$, and every $F_{i,j}, G_{i,j}$ and $H_{i,j}$,

$$\Big(\sigma'(\mathsf{L}), \sigma'(\mathsf{M}), \{\mathsf{NM.Decode}\big(F_{i,j}(\sigma_{i,j}(\mathsf{L})), G_{i,j}(\sigma_{i,j}(\mathsf{M})), H_{i,j}(\sigma_{i,j}(\mathsf{R}))\big)\}_{i,j} \Big| \mathsf{L}, \mathsf{M}, \mathsf{R} \leftarrow \mathsf{NM.Code}(\mathsf{m}_{i^*}^{j^*})\Big) \approx_\epsilon$$
$$\Big(\sigma'(\mathsf{L}), \sigma'(\mathsf{M}), \{\mathsf{NM.Decode}\big(F_{i,j}(\sigma_{i,j}(\mathsf{L})), G_{i,j}(\sigma_{i,j}(\mathsf{M})), H_{i,j}(\sigma_{i,j}(\mathsf{R}))\big)\}_{i,j} \Big| \mathsf{L}, \mathsf{M}, \mathsf{R} \leftarrow \mathsf{NM.Code}(0)\Big)$$

But these distributions upon post-processing (via the functions $w_{\mathsf{MIM}}, y_{\mathsf{MIM}}, z_{\mathsf{MIM}}$) exactly correspond to the outputs of $\mathsf{Hyb}_{3,i^*,j^*,k^*-1}$ and $\mathsf{Hyb}_{3,i^*,j^*,k^*}$ respectively, whenever $\widetilde{c}_{k^*}^{j^*}, \overline{c}_{k^*}^{j^*}$ and $\widehat{c}_{k^*}^{j^*}$ are all different. On the other hand, when any two of the three values $\widetilde{c}_{k^*}^{j^*}, \overline{c}_{k^*}^{j^*}$ and $\widehat{c}_{k^*}^{j^*}$ are identical, the distributions $\mathsf{Hyb}_{3,i^*,j^*,k^*-1}$ and $\mathsf{Hyb}_{3,i^*,j^*,k^*}$ are statistically indistinguishable because of the two-out-of-three secret sharing property of the code, i.e. they jointly do not depend on all three of the shares, L, R and M. Since $\epsilon(\lambda) = \mathsf{negl}(\lambda)$, this contradicts Equation (1), as desired. □

Finally, this proof also directly extends to demonstrate that the security of the watchlist protocol holds against sub-exponential adversaries that run in time less than or equal to T, where T denotes the running time of adversaries against which the underlying two-party computation protocol is 2-rewinding sender secure.

8 Three-Round Inner Protocol

In this section, we give a construction of a three-round inner protocol that makes black-box use of a two-round semi-malicious secure OT protocol. We start with the definition of the inner protocol.

8.1 Definition

We recall the syntax of the inner protocol from [IKSS21].

Syntax. The three-round inner protocol computing a function f is given by a tuple of algorithms $(\Pi_1, \Pi_2, \Pi_3, \mathsf{out}_\Pi)$ with the following syntax. For each round $r \in [3]$, 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 the public decoder $\mathsf{out}_\Pi(\pi(3))$ to compute the output.[11]

Definition 8.1 ([IKSS21])**.** *The protocol Π is said to be an inner protocol for computing a funtion f if it satisfies the following properties.*

- ***Correctness.*** *The protocol Π correctly computes a function f if for every choice of inputs x_i for party P_i,*
$$\Pr[\mathsf{out}_\Pi(\pi(3)) = f(x_1, \ldots, x_n)] = 1$$
where $\pi(3)$ denotes the transcript of the protocol Π when the input of P_i is x_i.

[11]By public output decoder, we mean that the decoder does not require access to private randomness of the parties to compute the output.

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

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

```
1. For each $i \in H$, compute $\pi_1^i := \Pi_1(1^\lambda, i, x_i; r_i)$.
2. Send $\{\pi_1^i\}_{i\in H}$ to $\mathcal{A}$.
3. Receive $\{\pi_1^i\}_{i\in M}$ from $\mathcal{A}$.
4. For each $i \in H$, compute $\pi_2^i := \Pi_2(1^\lambda, i, x_i, \pi(1); r_i)$.
5. Send $\{\pi_2^i\}_{i\in H}$ to $\mathcal{A}$.
6. Receive $\{\pi_2^i, (x_i, r_i)\}_{i\in M}$ from $\mathcal{A}$.
7. For each $i \in H$, compute $\pi_3^i := \Pi_3(1^\lambda, i, x_i, \pi(2); r_i)$.
8. Send $\{\pi_3^i\}_{i\in H}$ to $\mathcal{A}$.
9. Receive $\{\pi_3^i\}_{i\in M}$ from $\mathcal{A}$.
10. Output the view of $\mathcal{A}$ and $\mathsf{out}_\Pi(\pi(3))$.
```

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

```
1. For each $i \in H$, compute $(\pi_1^i, \mathsf{td}_i) := \mathsf{Sim}_\Pi(1^\lambda, i)$.
2. Send $\{\pi_1^i\}_{i\in H}$ to $\mathcal{A}$.
3. Receive $\{\pi_1^i\}_{i\in M}$ from $\mathcal{A}$.
4. For each $i \in H$, compute $\pi_2^i := \mathsf{Sim}_\Pi(1^\lambda, i, \pi(1))$.
5. Send $\{\pi_2^i\}_{i\in H}$ to $\mathcal{A}$.
6. Receive $\{\pi_2^i, (x_i, r_i)\}_{i\in M}$ from $\mathcal{A}$.
7. Check if the messages sent by corrupt parties in $\pi(2)$ are consistent with $\{x_i, r_i\}_{i\in M}$.
8. **Semi-Malicious Security:** If they are consistent:
    (a) For each $i \in H$, compute $\pi_3^i \leftarrow \mathsf{Sim}_\Pi(1^\lambda, i, f(x_1, \ldots, x_n), \{x_j, r_j\}_{j\in M}, \pi(2))$.
9. **Equivocality:** If they are not consistent:
    (a) For each $i \in H$, compute $\pi_3^i \leftarrow \mathsf{Sim}_\Pi(1^\lambda, i, \{x_i\}_{i\in H}, \pi(2))$.
10. Send $\{\pi_3^i\}_{i\in H}$ to $\mathcal{A}$.
11. Receive $\{\pi_3^i\}_{i\in M}$ from $\mathcal{A}$.
12. Output the view of $\mathcal{A}$ and $\mathsf{out}_\Pi(\pi(3))$.
```

Figure 8: Security Game for the Inner Protocol

The main theorem we prove in this section is:

Theorem 8.2. *Assume black-box access to a two-round semi-malicious secure OT protocol. Then, there exists a three-round inner protocol satisfying Definition 8.1.*

8.2 Building Blocks

The construction makes use of the following building blocks.

8.2.1 Protocol Γ

Γ is a three-round protocol that computes the double selection functionality DS [PS21] with publicly decodable transcript. The double selection functionality is a multiparty functionality where P_1 has input $(\alpha, r) \in \{0,1\} \times \{0,1\}$, P_2 has input $(s_0, s_1) \in \{0,1\} \times \{0,1\}$ and for each $i \in [n]$, P_i has additional inputs (z_0^i, z_1^i). The output of the functionality is given by $(s_\alpha \oplus r, \{z^i_{s_{\alpha \oplus r}}\}_{i \in [n]})$. The protocol Γ has similar syntax as that of Π described earlier. Apart from standard correctness, we need this protocol to satisfy the following security property.

Security of protocol Γ. Let x_i be the input used by party P_i in the protocol Γ. Let $\mathcal{A}$ be any malicious adversary that is corrupting a set of parties indexed by M and let H be the set of honest parties. We require the existence of stateful Sim_Γ such that:

- Sim_Γ on $1^\lambda, i, \gamma(0)$ (where $\gamma(0)$ is the null string) outputs the first round message γ_1^i and td_i. Using td_i and the input x_i of party P_i, there is a polynomial time algorithm $\mathsf{Equivocate}$ that outputs the second and third round messages of the protocol Γ such that the view of any adversary in the real execution with the honest parties is computationally indistinguishable to the distribution of messages generated as above.

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

  where the real and ideal experiments are described in Figure 8 and for each $i \in H$, r_i is uniformly chosen. Note that the only difference between this game and the one described in Figure 8 is that in this game, the adversary is forced to reveal the input and randomness after she sends the first round message, whereas in the other game, she only reveals this information after sending the second round message.

Construction. We give the formal description of the protocol Γ in Figure 10 which is adapted from the work of Patra and Srinivasan [PS21]. The only difference between our construction and their construction is that we make use of a two-round oblivious transfer protocol with equivocal receiver security [GS18, IKSS21]. From Theorem 3.4, we note that such an OT protocol can be constructed from black-box use of a two-message semi-malicious OT protocol.

Sketch of Security. Sim_Γ runs the equivocal simulator for the oblivious transfer protocol and obtains the first round message along with the trapdoor. It sends this first round message to the adversary. If the first round message or the second round message generated by the adversary is inconsistent with the sent input, randomness pair, then Sim_Γ uses the above trapdoor and the inputs of the honest parties to derive the appropriate secret keys and send the honestly computed second and third round messages. On the other hand, if the messages are consistent, then Sim_Γ uses the same simulation strategy as given in [PS21].

8.2.2 Protocol Ψ

The protocol Ψ is a black-box, publicly decodable, two-round MPC protocol for computing arbitrary functions in the OT correlations model. Such a protocol was constructed in [GIS18]. We need a couple of additional properties from this protocol Ψ.

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

1. For each $i \in H$, compute $\gamma_1^i := \Gamma_1(1^\lambda, i, x_i; r_i)$.
2. Send $\{\gamma_1^i\}_{i \in H}$ to $\mathcal{A}$.
3. Receive $\{\gamma_1^i, (x_i, r_i)\}_{i \in M}$ from $\mathcal{A}$.
4. For round $r \in \{2, 3\}$:
    (a) For each $i \in H$, compute $\gamma_r^i := \Gamma_r(1^\lambda, i, x_i, \gamma(r-1); r_i)$.
    (b) Send $\{\gamma_r^i\}_{i \in H}$ to $\mathcal{A}$.
    (c) Receive $\{\gamma_r^i\}_{i \in M}$ from $\mathcal{A}$.
5. Output the view of $\mathcal{A}$ and $\mathsf{out}_\Gamma(\gamma(3))$.
```

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

1. For each $i \in H$, compute $\gamma_1^i := \mathsf{Sim}_\Gamma(1^\lambda, i)$.
2. Send $\{\gamma_1^i\}_{i \in H}$ to $\mathcal{A}$.
3. Receive $\{\gamma_1^i, (x_i, r_i)\}_{i \in M}$ from $\mathcal{A}$.
4. For round $r \in \{2, 3\}$:
    (a) Check if the messages sent by corrupt parties in $\gamma(r-1)$ are consistent with $\{x_i, r_i\}_{i \in M}$.
    (b) Semi-Malicious Security: If they are consistent:
        i. For each $i \in H$, compute $\gamma_r^i := \mathsf{Sim}_\Gamma(1^\lambda, i, \{x_i, r_i\}_{i \in M}, \mathsf{DS}(x_1, \ldots, x_n), \gamma(r-1))$.
    (c) Equivocality: If they are not consistent:
        i. For each $i \in H$, compute $\Gamma_r^i := \mathsf{Sim}_\Gamma(1^\lambda, i, \{x_i\}_{i \in H}, \gamma(r-1))$.
    (d) Send $\{\gamma_r^i\}_{i \in H}$ to $\mathcal{A}$.
    (e) Receive $\{\gamma_r^i\}_{i \in M}$ from $\mathcal{A}$.
5. Output the view of $\mathcal{A}$ and $\mathsf{out}_\Gamma(\gamma(3))$.
```

Figure 9: Security Game for the Protocol Γ

1. The protocol Ψ satisfies the same security properties as that of Γ. Specifically, if the first round message sent by the adversarial parties is consistent with the provided input and randomness, then the only information that the adversary learns is the output of the functionality. That is, the simulator produces the last round message of the protocol given the adversarial inputs, randomness and the output of the function such that this is indistinguishable to the real execution. On the other hand, if the input, randomness pair provided by the adversary are inconsistent, then we require the simulator to generate the last round message of the protocol given the honest party's inputs such that the view of the adversary generated above is indistinguishable to the view of the adversary in the real execution. [IKSS21] noted that if the oblivious transfer in [GIS18] is replaced with a OT with equivocal receiver security then [GIS18] already satisfies this property.

2. We assume that Ψ has the following syntactic form. For each of the OT correlation that needs to be generated between a particular receiver and a sender, the receiver samples random bits (α, r), the sender samples random bits s_0, s_1 and every party (including the sender and the receiver) samples two random strings z_0^i, z_1^i. To generate the first round and the second round message in the protocol Ψ only the above sampled information is needed. However, to compute the output, every party requires $(s_\alpha \oplus r, \{z^i_{s_\alpha \oplus r}\}_{i \in [n]})$. For security of Ψ, we need to ensure that only this information is revealed. Specifically, for each OT correlation generated between every pair of parties, the simulator for Ψ takes the concatenation of these outputs and simulates the second round message on behalf of the honest parties. The proof that [GIS18]

construction satisfies this property is implicit in [PS21, Theorem 5.22].

- P_1 has input bits (α, r) and strings (z_0^1, z_1^1), P_2 has input bits s_0, s_1 and strings (z_0^2, z_1^2), and each P_i for $i \in [3, n]$ has input strings (z_0^i, z_1^i). The output of the functionality is given by $(s_{\alpha \oplus r}, \{z^i_{s_\alpha \oplus r}\}_{i \in [n]})$.
- **Round-1:** In the first round,
  1. P_1 computes $(\mathsf{otr}, sk) \leftarrow \mathsf{OT}_1(1^\lambda, \alpha)$ and $(\mathsf{otr}', sk') \leftarrow \mathsf{OT}_1(1^\lambda, r)$.
  2. P_2 computes $(\mathsf{otr}_b, sk_b) \leftarrow \mathsf{OT}_1(1^\lambda, s_b)$ for each $b \in \{0, 1\}$.
  3. For each $i \in [n]$, P_i computes $(\mathsf{otr}_b^i, sk_b^i) \leftarrow \mathsf{OT}_1(1^\lambda, z_b^i)$ for each $b \in \{0, 1\}$ where we compute OT_1 on each bit of z_b^i.
  4. The parties broadcast the above computed messages.
- **Round-2:** In the second round,
  1. For each $i \in [n]$,
     (a) P_i chooses a masking string m^i uniformly at random.
     (b) It computes $\mathsf{ots}^i \leftarrow \mathsf{OT}_2(\mathsf{otr}, \mathsf{OT}_2(\mathsf{otr}_0, z_0^i \oplus m^i, z_1^i \oplus m^i), \mathsf{OT}_2(\mathsf{otr}_1, z_0^i \oplus m^i, z_1^i \oplus m^i))$.
     (c) It computes $\mathsf{ots}'^i \leftarrow \mathsf{OT}_2(\mathsf{otr}', m^i, z_0^i \oplus z_1^i \oplus m^i)$.
  2. P_2 additionally computes $\overline{\mathsf{ots}} \leftarrow \mathsf{OT}_2(\mathsf{otr}, (s_0, sk_0), (s_1, sk_1))$
  3. Each party broadcasts the above computed messages to the other parties.
- **Round-3:** In the third round, P_1 does the following:
  1. It first recovers (s_α, sk_α) from $\overline{\mathsf{ots}}$ and sk.
  2. For each $i \in [n]$:
     (a) It then recovers $\mathsf{OT}_2(\mathsf{otr}_\alpha, z_0^i \oplus m^i, z_1^i \oplus m^i)$ from ots^i and sk.
     (b) It then computes $(z_0^i \oplus z_1^i)(s_\alpha) \oplus z_0^i \oplus m^i$ from $\mathsf{OT}_2(\mathsf{otr}_\alpha, z_0^i \oplus m^i, z_1^i \oplus m^i)$ and sk_α.
     (c) It then computes $(z_0^i \oplus z_1^i)r \oplus m^i$ from ots'^i and sk'.
     (d) It finally computes $z^i_{s_\alpha \oplus r}$ by XORin the values computed in the last two steps.
  3. It broadcasts $(s_\alpha \oplus r, \{z^i_{s_\alpha \oplus r}\}_{i \in [n]})$ to every party and they output the received value.

Figure 10: Description of the Protocol Γ adapted from [PS21]

8.3 Construction

We give the formal description of the construction in Figure 11.

8.4 Proof of Security

In this subsection, we show that the protocol Π described in Figure 11 satisfies the Definition 8.1. We start with the description of the simulator Sim_Π.

8.4.1 Description of Simulator

Let $\mathcal{A}$ be an adversary that is corrupting a set of parties indexed by M and let H be the set of honest parties.

- **Round-1:** For each OT correlation to be generated between P_i (acting as the receiver) and P_j (acting as the sender):
    1. P_i chooses random bits α, r.
    2. P_j chooses two bits s_0, s_1.
    3. For each $k \in [n]$, P_k chooses two random strings z_0^k, z_1^k.
    4. The parties generate the first round message of the protocol Γ using the above sampled inputs.
- **Round-2:** For each $i \in [n]$:
    1. P_i generates the second round message for each execution of the Γ protocol initiated in the first round.
    2. Let y_i be the concatenation of the all the inputs chosen by P_i in each execution of the Γ protocol.
    3. Let x_i denote the augmented inputs that includes P_i's private input χ_i along with y_i.
    4. P_i computes $\psi_1^i \leftarrow \Psi_1(1^\lambda, i, x_i)$.
    5. P_i broadcasts ψ_1^i to other parties along with the second round messages of all the executions of the Γ protocol.
- **Round-3:** For each $i \in [n]$:
    1. P_i generates the third round message of all the executions of the Γ protocol.
    2. P_i computes $\psi_2^i \leftarrow \Psi_1(1^\lambda, i, x_i, \psi(1))$ where $\psi(r)$ is the transcript in the protocol Ψ generated in the first r rounds.
    3. P_i broadcasts ψ_2^i along with the third round messages in all the executions of the Γ protocol
- **Output Computation:** For each $i \in [n]$:
    1. P_i computes the output of all the executions of the Γ protocol and let δ denote the concatenation of all these outputs.
    2. P_i outputs $\mathsf{out}_\Psi(\psi(2), \delta)$.

Figure 11: Three-Round Inner Protocol

- **Round-1 Message from Sim_Π and $\mathcal{A}$.** For each execution of the Γ protocol that is initiated in the first round, Sim_Π samples uniform random inputs on behalf of the honest parties and runs each execution honestly using the above sampled inputs. Sim_Π receives the first round message from $\mathcal{A}$.

- **Round-2 Message from Sim_Π and $\mathcal{A}$.** Sim_Π generates the second round message of each Γ execution honestly but uses the simulator Sim_Ψ to generate the the first round message in the protocol Ψ on behalf of the honest parties. It receives the second round message from $\mathcal{A}$ along with the input and randomness used by the adversary.

- **Round-3 Message from Sim_Π and $\mathcal{A}$.** If the input, randomness pair provided by the adversary are inconsistent, then Sim_Π receives the inputs $\{\chi_i\}_{i \in H}$ of the honest parties and it runs Sim_Ψ on $\{(\chi_i, y_i)\}_{i \in H}$ where y_i is the concatenation of all the inputs used by P_i in executions of the Γ protocol to obtain the final round message in Ψ. If the inputs, randomness pair are consistent, then Sim_Π extracts the outputs that the adversary receives in each execution of the Γ protocol (denoted by δ) from the provided randomness and the inputs used by the honest parties. It then runs Sim_Ψ on the input, randomness pair of the malicious parties, δ and the output $f(\chi_1, \ldots, \chi_n)$ and obtains the final round message of Ψ. Sim_Π sends the final round message to adversary and receives the final round message generated by $\mathcal{A}$ on behalf of the corrupted parties.

8.4.2 Proof of Indistinguishability

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the output of $\mathsf{Real}(\mathcal{A}, \{\chi_i, r_i\}_{i \in H})$.
- $\underline{\mathsf{Hyb}_1 :}$ In this hybrid, we make the following changes:

  1. We generate the first round message from each $i \in H$ and for each execution of the protocol Γ as $(\gamma_1^i, \mathsf{td}_i) \leftarrow \mathsf{Sim}_\Gamma(1^\lambda, i, \gamma(0))$. We then generate the second and the third round messages using $\mathsf{Equivocate}$.

  It follows from the first security property of the protocol Γ that this hybrid is computationally indistinguishable to Hyb_0.

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

  1. We generate the first round message of the protocol $\{\gamma_1^i\}_{i \in H}$ on behalf of the honest parties as before and we obtain the first round message from the adversary. We run in super-polynomial time and extract the input and randomness used to generate the first round message. We provide this to Sim_Γ and use it to generate the second and third round message as in the Ideal_Γ experiment.

  In Lemma 8.3, we show that Hyb_2 is computationally indistinguishable to Hyb_1.

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

  1. We generate the first round and the second round messages in the protocol Ψ as described in the simulation.

  It directly follows from the security property of Ψ that this hybrid is computationally indistinguishable to the previous hybrid.

- $\underline{\mathsf{Hyb}_4 - \mathsf{Hyb}_5 :}$ In this hybrid, we reverse the changes made in Hyb_2 and Hyb_1 respectively. Note that Hyb_5 is identical to the Ideal experiment generated using Sim_Π.

Lemma 8.3. *Assuming the security properties of the protocol Γ (described in Figure 9), we have $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$.*

Proof. Assume for the sake of contradiction that Hyb_2 and Hyb_1 are computationally distinguishable with non-negligible advantage. We show that this contradicts the security of the protocol Γ. To show this, we modify the Real_Γ experiment such that the messages in this experiment are generated as in Hyb_1. We note that it follows from the first security property of Γ that the original Real_Γ experiment is computationally indistinguishable to the modified Real_Γ experiment.

Since the first round messages in both Hyb_2 and Hyb_1 for each execution of the protocol Γ are identically distributed to the output of Sim_Γ, we can non-uniformly fix this message. As a result, we can also non-uniformly fix the first round message from the adversary. As a result of this fixing, we can get the input and randomness used by the adversary to generate its first round message as part of non-uniform advice.

We provide the external challenger with the input and randomness used by the adversary to generate the first round message. We use the messages generated by the challenger to complete the execution with the adversary. We run the distinguisher between Hyb_2 and Hyb_1 on the view of the adversary and the output of the honest parties and we output whatever the distinguisher outputs.

If the messages generated by the external challenger in the second and third rounds of the protocol Γ is distributed as in the modified Real_Γ experiment, then the input to the distinguisher is identical to Hyb_1. Otherwise, if these messages are generated by challenger as in the Ideal_Γ experiment, then input to the distinguisher is identically distributed to Hyb_2. Thus, if the distinguisher can distinguish between Hyb_2 and Hyb_3 with non-negligible advantage, then this contradicts the security property of the protocol Γ. □

9 4-Round Black-Box MPC Protocol

In this section, we give our construction of a four-round black-box MPC protocol from any two-message OT protocol that has super-polynomial time security against malicious receivers and sub-exponential indistinguishability-based security against malicious sendersß. Specifically, we prove the following theorem.

Theorem 9.1. *For some $\epsilon > 0$, assume black-box access to a two-round oblivious transfer protocol with super-polynomial time simulation security against malicious receivers and $(2^{\lambda^\epsilon}, 2^{-\lambda^\epsilon})$-indistinguishability-based security against malicious senders. Then, there exists a four-round protocol for computing general functions.*

9.1 Building Blocks

The construction makes use of the following building blocks:

1. A three-round watchlist protocol $\mathsf{WL} = (\mathsf{WL}_1, \mathsf{WL}_2, \mathsf{WL}_3, \mathsf{out}_{\mathsf{WL}})$ satisfying Definition 7.1 with $k = \lambda$, $\ell = 6\lambda n^2 + 1$. Let $T_1(\lambda)$ (abbreviated as T_1) be the running time of $\mathsf{Sim}_{\mathsf{WL}}$ (which is the SPS simulator for the watchlist protocol). Let $T_2(\lambda)$ (abbreviated as T_2) to be the running time of $\mathsf{Sim}_{\mathsf{WL},R}$ (which is the special SPS extractor that over extracts the receiver inputs).

2. A two-round n-client, m-server MPC protocol $\Phi = (\Phi_1, \Phi_2, \mathsf{out}_\Phi)$ that computes an augmented functionality $g(\cdot)$ and satisfies $((T_1 + T_2) \cdot \mathsf{poly}(\cdot), \mathsf{negl})$-privacy with knowledge of outputs property against any adversary corrupting upto t servers and an arbitrary number of clients. Let us now give the description of the functionality g. g takes in (χ_i, k_i) from each party P_i where k_i is a MAC key. It computes $y = f(\chi_1, \ldots, \chi_n)$ and computes $\sigma_i = \mathsf{MAC}(k_i, y)$ for each $i \in [n]$. It outputs $(y, \sigma_1, \ldots, \sigma_n)$. We call this protocol as the outer protocol. We set $t = 2\lambda n^2$ and $m = 3t + 1$. We need this protocol to additionally satisfy the property that the first round message generated by the simulator on behalf of the honest clients to the corrupted servers is identically distributed to the first round messages generated by honest clients on some default input. We note that [IKP10, Pas12] constructed such a protocol making black-box use of a $((T_1 + T_2) \cdot \mathsf{poly}(\cdot), \mathsf{negl})$-secure PRG. As noted in [IKSS21], we can delegate the

PRG computations done by the servers to the clients and ensure that the computations done by the servers are information-theoretic.

3. For each $h \in [m]$, a three-round inner protocol $\Pi_h = (\Pi_{h,1}, \Pi_{h,2}, \Pi_{h,3}, \mathsf{out}_{\Pi_h})$ for computing the functionality of the h-th server in the outer protocol. We require this protocol to satisfy Definition 8.1 against adversaries running in time $(T_1 + T_2) \cdot \mathsf{poly}(\lambda)$ and the distinguishing advantage being $\mathsf{negl}(\lambda)$.

9.2 Construction

We give the description of the protocol in Figure 12.

9.3 Proof of Security

In this subsection, we prove that the MPC protocol described in Figure 12 satisfies the standard simulation-based security definition. We start with the description of the simulator.

9.3.1 Description of Simulator

Let $\mathcal{A}$ be an adversary that is corrupting a subset $M \subset [n]$ of the clients and let H be the set of honest clients. Sim does the following:

- **Round-1 Message from Sim and $\mathcal{A}$:**
  1. For each $i \in H$:
     (a) It computes $(\phi_1^{i \to 1}, \ldots, \phi_1^{i \to m}) \leftarrow \Phi_1(1^\lambda, i, \mathbf{0})$ where $\mathbf{0}$ is a default input.
     (b) Sim chooses a random subset $K_i \subset [m]$ of size λ and sets $x_{i,j} = K_i$, for every $j \in [n] \setminus \{i\}$. It then 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 $j \in [n] \setminus \{i\}$.
     (c) It computes $\mathsf{wl}_1^i \leftarrow \mathsf{WL}_1(1^\lambda, i, \{x_{i,j}, y_{i,j}\}_{j \in [n] \setminus \{i\}})$.
  2. It sends $\{\mathsf{wl}_1^i\}_{i \in H}$ to the adversary.
  3. Sim receives the first round message $\{\mathsf{wl}_1^i\}_{i \in M}$ from the adversary.

- **Extracting the watchlist inputs used by $\mathcal{A}$.** Sim does the following:
  1. It generates the second and the third round of the protocol using the input $\{\phi_1^{i \to h}\}_{i \in H, h \in [m]}$ and the randomness $\{r_{i,h}\}_{i \in H, h \in [m]}$ honestly as described in Figure 12.
  2. If the adversary aborts during this phase, or sends an invalid message then Sim sends $\perp$ to the ideal functionality and outputs the view of $\mathcal{A}$.
  3. Otherwise, Sim runs the extractor $\mathsf{Ext}_{\mathsf{WL},R}$ and $\mathsf{Ext}_{\mathsf{WL},S}$ on the adversarial messages while generating the honest parties messages as explained before to obtain $\{x_{i,j}, y_{i,j}\}_{i \in M, j \in H}$.[12]
  4. Sim also estimates the probability that $\mathcal{A}$ does not abort during this phase or does not send an invalid message. For this purpose, it repeatedly rewinds the second and third round of the protocol until it obtains 12λ executions where the adversary has not aborted

[12]Specifically, in order to run $\mathsf{Ext}_{\mathsf{WL},\mathcal{R}}$, Sim rewinds the adversary until it obtains k accepting transcripts.

- **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. It runs $\mathsf{out}_{\mathsf{WL}}$ on i, $\{x_{i,j}, y_{i,j}\}_{j\in[n]\setminus\{i\}}$, the random tape used to generate the messages in WL and $\mathsf{wl}(3)$ 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 $\ell \in [2]$, 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 above checks fail, then it aborts.
  4. Else, for each $h \in [m]$, it computes $\pi_{h,3}^i := \Pi_{h,3}(1^\lambda, i, \phi_1^{i\to h}, \pi_h(2); r_{i,h})$.
  5. It broadcasts $\{\pi_{h,3}^i\}_{h\in[m]}$ to every party.
- **Output Computation.** To compute the output, P_i does the following:
  1. If a party has aborted before sending the fourth round message, output $\perp$.
  2. For every $h \in [m]$, it computes $\phi_2^h := \mathsf{out}_{\Pi_h}(i, \pi_h(3))$.
  3. It runs out_Φ on $(\{\phi_2^h\}_{h\in[m]})$ 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.

Figure 12: Description of the Four-Round MPC Protocol

nor sent an invalid message. Let T be the total number of trials needed until this event happens. Sim sets $\widetilde{\epsilon} = \frac{12\lambda}{T}$. Sim has an internal step counter and if it runs for more than 2^λ steps, it aborts.

- **Round-2 and Round-3 messages from Sim and $\mathcal{A}$.** Sim repeats the following for $\frac{\lambda^2}{\epsilon}$ time:

  1. For each $i \in M$ and $j \in H$, it parses the above extracted $x_{i,j}$ as a subset of $[m]$ of size λ and it sets $K = \cup_{i \in M, j \in H} x_{i,j}$.
  2. For each $h \in K$, Sim generates the messages in the protocol Π_h using the input $\phi_1^{i \to h}$ and randomness $r_{i,h}$ for each $i \in H$.
  3. For each $h \notin K$, Sim generates the messages in Π_h using the simulator Sim_{Π_h}.
  4. Sim receives the second and third round messages from $\mathcal{A}$. If $\mathcal{A}$ aborts during this phase, or sends an invalid message, then Sim goes to the next iteration.

  If each of $\frac{\lambda^2}{\epsilon}$ iterations fail, then Sim outputs a special symbol ABORT.

- **Finding the set of Inconsistent Executions.** Sim does the following:

  1. It initializes an empty set C.
  2. It adds an $h \in [m]$ to C if there exists at least one $i \in M$ such that for each $j \in H$, the input, randomness pair present in $y_{i,j}$ is inconsistent with the messages generated by P_i in Π_h, or if the PRG computations are incorrect.
  3. If $|C| > \lambda \cdot n^2$, Sim aborts the execution and instructs all the honest parties to output $\perp$.

- **Round-4 Message from Sim and $\mathcal{A}$.**

  1. Sim performs the same checks that an honest party does at the end of the third round. If any of the checks done by an honest party fails, then Sim instructs the ideal functionality to send $\perp$ to that specific honest party.
  2. Sim sends some inconsistent input, randomness pair on behalf of the malicious parties for each $h \in C$ to Sim_{Π_h} and provides $\{\phi_1^{i \to h}\}_{i \in H}$ as the corresponding private inputs of the honest parties. It obtains the final round message to be sent by all the honest parties from Sim_{Π_h}.
  3. Sim starts running the simulator Sim_Φ for the protocol Φ by corrupting the set of clients indexed by M and the set of serves indexed by $K \cup C$. It provides $\{\phi_1^{i \to h}\}_{i \in H, h \in K \cup C}$ as the dummy first round messages generated by the honest clients to the corrupted servers.
  4. For each $h \notin K \cup C$, let $\{(r_{i,h}, \phi_1^{i \to h})\}_{i \in M}$ be the randomness, input pair that is consistent with the messages in Π_h generated by $\mathcal{A}$ such that the PRG computations in $\{\phi_1^{i \to h}\}_{i \in M}$ are correct. Sim provides $\{\phi_1^{i \to h}\}_{i \in M, h \notin K \cup C}$ to Sim_Φ as the first round messages sent by the malicious clients to the honest servers. Sim_Φ queries the ideal functionality on $\{z_i\}_{i \in M}$.
  5. Sim parses z_i as (χ_i, k_i) for each $i \in M$. It queries its trusted functionality on $\{\chi_i\}_{i \in M}$ and obtains the output y. For each $i \in M$, Sim computes $\sigma_i = \mathsf{MAC}(k_i, y)$ and for each $i \in H$, it chooses σ_i uniformly. It provides $(y, \sigma_1, \ldots, \sigma_n)$ to Sim_Φ. Sim_Φ generates the final round message $\{\phi_2^h\}_{h \notin K \cup C}$.
  6. For each $h \notin K \cup C$, Sim provides Sim_{Π_h} with the malicious clients randomness and inputs $\{(r_{i,h}, \phi_1^{i \to h})\}_{i \in M}$ and the output ϕ_2^h for the inner protocol and obtains the final round message sent by the honest parties.

7. Sim sends the last round message on behalf of all the unaborted parties to $\mathcal{A}$ and receives the final round message.

- **Output Computation.**

1. If some party has aborted at the end of the third round, then Sim instructs the ideal functionality to send $\perp$ to all the honest parties.
2. For each $h \in [m]$, Sim computes ϕ_2^h using out_{Π_h}.
3. It then runs out_Φ on $(\phi_2^1, \ldots, \phi_2^m)$ to compute $(y', \sigma_1', \ldots, \sigma_n')$.
4. For each $i \in H$, if $(y, \sigma_i) \neq (y', \sigma_i')$ then Sim instructs the trusted functionality to send $\perp$ to P_i. Else, it instructs the functionality to deliver the output to P_i.

Running time analysis of Sim. Let ϵ be the probability that the adversary does not abort nor sends an invalid message in the step where Sim is extracting the watchlist inputs. By a standard argument (see for instance [GK96a]), we can show that $\widetilde{\epsilon}$ is at most a constant factor of the actual ϵ except with $2^{-\lambda}$ probability. The expected running time of Sim consists of the following components:

- All the steps before the extraction of the watchlist inputs used by $\mathcal{A}$. The expected running time of these steps is $\mathsf{poly}(\lambda)$.

- With probability $1 - \epsilon$, Sim outputs the view of $\mathcal{A}$ in $\mathsf{poly}(\lambda)$ steps.

- With probability ϵ, the expected additional work done by Sim is given by $\frac{12\lambda}{\epsilon}\mathsf{poly}(\lambda) + \frac{\lambda^2}{\epsilon}\mathsf{poly}(\lambda) + \mathsf{poly}(\lambda) \cdot (E_{\mathsf{WL},R} + E_{\mathsf{WL},S})$ where $E_{\mathsf{WL},R}$ is the expected running time of $\mathsf{Ext}_{\mathsf{WL},R}$ and $E_{\mathsf{WL},S}$ is the expected running time of $\mathsf{Ext}_{\mathsf{WL},S}$.

- In case where $\widetilde{\epsilon}$ is not within a constant factor of ϵ, we can bound the running time of Sim by 2^λ. But the probability of this event happening is $2^{-\lambda}$.

Thus, the expected running time of Sim is upper bounded by $\mathsf{poly}(\lambda)$ as $\epsilon \cdot E_{\mathsf{WL},R}$ and $\epsilon \cdot E_{\mathsf{WL},S}$ is bounded above by $\mathsf{poly}(\lambda)$. Specifically, $E_{\mathsf{WL},R}$ is upper bounded by $\frac{k}{\epsilon} \cdot \mathsf{poly}(\lambda)$ and $E_{\mathsf{WL},S}$ is upper bounded by $\frac{\mathsf{poly}(\lambda)}{\epsilon}$.

9.3.2 Proof of Indistinguishability

We now show that the real execution and the ideal execution generated by Sim are computationally indistinguishable using 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:

1. We define an adversary $\mathcal{A}_1$ that internally interacts with $\mathcal{A}$. $\mathcal{A}_1$ obtains the messages corresponding to the watchlist protocol externally. It generates the messages for the protocols $\{\Pi_h\}_{h \in [m]}$ as in Hyb_0. It combines these two messages and sends them to $\mathcal{A}$. Whenever $\mathcal{A}_1$ receives a message from $\mathcal{A}$, it extracts the messages corresponding to the watchlist protocol and forwards them externally.

2. We run the simulator for the watchlist protocol $\mathsf{Sim}_{\mathsf{WL}}$ on $\mathcal{A}_1$. $\mathsf{Sim}_{\mathsf{WL}}$ queries the ideal functionality on the inputs of the adversary used in the watchlist protocol and we answer the query using the inputs $\{(x_{i,j}, y_{i,j})\}_{i\in H, j\in M}$ of the honest parties.
3. $\mathsf{Sim}_{\mathsf{WL}}$ outputs the view of $\mathcal{A}_1$, $\{(x_{i,j}, y_{i,j})\}_{i\in M, j\in H}$ along with the instruction for each $i \in H$ to either deliver the output of the watchlist functionality to P_i or deliver $\perp$. We compute the outputs of the honest unaborted parties using their inputs used in the watchlist protocol and the corrupted parties inputs output by $\mathsf{Sim}_{\mathsf{WL}}$. We extract the view of $\mathcal{A}$ in the first three rounds from view of $\mathcal{A}_1$.
4. For the unaborted parties, we use the output of the watchlist functionality to perform the same checks as described in Figure 12 before sending the fourth round message. We then generate the final round inner protocol message $\pi^i_{h,3}$ for each $h \in [m]$ and generate the view of the $\mathcal{A}$. We compute the output of each honest party as described in the protocol. We output the final view of $\mathcal{A}$ and the output of all the honest parties.

In Lemma 9.2, we show that Hyb_0 and Hyb_1 are computationally indistinguishable from the security of the watchlist protocol.

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

1. We define an adversary $\mathcal{A}_2$ that internally interacts with $\mathcal{A}$. We start running $\mathsf{Sim}_{\mathsf{WL}}$ on $\mathcal{A}_2$ and now explain how $\mathcal{A}_2$ generates the messages corresponding to the inner protocol.
2. We run $\mathsf{Sim}_{\mathsf{WL}}$ on $\mathcal{A}_2$ to generate the first round message of the watchlist protocol on behalf of the honest parties. We send this to $\mathcal{A}_2$ which forwards it to $\mathcal{A}$. It receives the first round message from $\mathcal{A}$. We run $\mathsf{Sim}_{\mathsf{WL}}$ on this message to extract $\{x_{i,j}\}_{i\in M, j\in H}$. We parse each $x_{i,j}$ as a subset of $[m]$ of size λ and denote $K = \cup_{i\in M, j\in H} x_{i,j}$.
3. For each $h \in K$, $\mathcal{A}_2$ generates the messages in the protocol Π_h as before using the honest parties inputs and uniformly chosen randomness. For each $h \notin K$, $\mathcal{A}_2$ uses the simulator Sim_{Π_h} to generate the messages on behalf of the honest parties.
4. $\mathsf{Sim}_{\mathsf{WL}}$ outputs the view of $\mathcal{A}_2$, $\{y_{i,j}\}_{i\in M, j\in H}$ along with the instruction for each $i \in H$ to either deliver the output of the watchlist functionality to P_i or deliver $\perp$. We compute the outputs of the honest unaborted parties in the watchlist protocol using their corresponding inputs used in this protocol. We extract the view of $\mathcal{A}$ in the first three rounds from view of $\mathcal{A}_2$.
5. For each $i \in M$ and $j \in H$, we parse $y_{i,j}$ as $(r_{i,h}, \phi_1^{i\to h})$ and check if the messages in the protocol Π_h sent by P_i are consistent with this input, randomness pair and if the PRG computations in $\phi_1^{i\to h}$ are correct. If for each $j \in H$, if this pair is inconsistent, or if the PRG computations in $\phi_1^{i\to h}$ are incorrect then we provide Sim_{Π_h} with some inconsistent input, randomness pair on behalf of the malicious parties and provide $\{\phi_1^{j\to h}\}_{j\in H}$ as the honest party inputs.
6. If for each $i \in M$, there exists at least one $j \in H$ such that the input randomness pair $(r_{i,h}, \phi_1^{i\to h})$ in $y_{i,j}$ is consistent with all the messages sent in Π_h and if the PRG computations in $\phi_1^{i\to h}$ are correct, then we provide $\phi_2^h := \Phi_2(i, \{\phi_1^{i\to h}\}_{i\in[n]})$ as the output of the functionality along with the consistent input, randomness pair of the corrupted parties to Sim_{Π_h} and obtain the final round message from Sim_{Π_h} on behalf of all unaborted

honest parties. We use this to generate the view of $\mathcal{A}$ as well as compute the output of all the honest parties as in the previous hybrid.

Let T_1 be the running time of $\mathsf{Sim}_{\mathsf{WL}}$. In Lemma 9.3, we show that $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$ from the security of the inner protocol against adversaries running in time T_1.

- $\underline{\mathsf{Hyb}_3 :}$ In this hybrid, we make the following changes:
  1. We initialize an empty set C.
  2. We add an $h \in [m]$ to C if there exists at least one $i \in M$ such that for each $j \in H$, the input, randomness pair present in $y_{i,j} = (r_{i,h}, \phi_1^{i \to h})$ is inconsistent with the messages generated by P_i in Π_h or if the PRG computations in $\phi_1^{i \to h}$ are incorrect.
  3. If $|C| > \lambda \cdot n^2$, we abort the execution and instruct all the honest parties to output $\bot$.

In Lemma 9.4, we show that $\mathsf{Hyb}_2 \approx_s \mathsf{Hyb}_3$.

- $\underline{\mathsf{Hyb}_4 :}$ In this hybrid, we make the following changes:
  1. We define an adversary $\mathcal{A}_3$ that internally interacts with $\mathcal{A}$. We start running $\mathsf{Sim}_{\mathsf{WL}}$ on $\mathcal{A}_3$ and now explain how $\mathcal{A}_3$ generates the messages corresponding to the inner protocol.
  2. We run $\mathsf{Sim}_{\mathsf{WL}}$ on $\mathcal{A}_3$ to generate the first round message of the watchlist protocol on behalf of the honest parties. We send this to $\mathcal{A}_3$ which forwards it to $\mathcal{A}$. It receives the first round message from $\mathcal{A}$. We run $\mathsf{Sim}_{\mathsf{WL}}$ on this message to extract $\{x_{i,j}\}_{i \in M, j \in H}$. We parse each $x_{i,j}$ as a subset of $[m]$ of size λ and denote $K = \cup_{i \in M, j \in H} x_{i,j}$.
  3. $\mathcal{A}_3$ starts running the simulator Sim_{Φ} for the outer protocol by corrupting the set of clients indexed by M and the set of servers indexed by K. Sim_{Φ} provides the first round messages $\{\phi_1^{i \to h}\}_{i \in H, h \in K}$ as the first round messages sent by the honest clients to the corrupted servers. It uses this as the honest party's input and chooses uniform randomness to generate the protocol messages in Π_h for each $h \in K$. It also uses this to answer the oracle query of $\mathsf{Sim}_{\mathsf{WL}}$. For each $h \notin K$, $\mathcal{A}_3$ uses the simulator Sim_{Π_h} to generate the messages in the protocol Π_h just like $\mathcal{A}_2$.
  4. $\mathsf{Sim}_{\mathsf{WL}}$ outputs the view of $\mathcal{A}_3$, $\{y_{i,j}\}_{i \in M, j \in H}$ along with the instruction for each $i \in H$ to either obtain the output of the watchlist functionality to P_i or obtain $\bot$. We compute the outputs of the honest unaborted parties using their inputs used in the watchlist protocol. We extract the view of $\mathcal{A}$ in the first three rounds from view of $\mathcal{A}_3$.
  5. At the end of the third round, we define the set C as in the previous hybrid and abort if $|C| > \lambda \cdot n^2$. If $|C| \le \lambda \cdot n^2$, we instruct Sim_{Φ} to adaptively corrupt the set of servers indexed by C and obtain $\{\phi_1^{i \to h}\}_{i \in H, h \in C}$ as the first round messages that honest clients send to the set of servers in C in the first round. We give this as the honest parties' inputs to the inner protocol simulator Sim_{Π_h} for each $h \in C$.
  6. For each $h \notin C \cup K$ and for each $i \in M$, we first extract the consistent input $\phi_1^{i \to h}$ (such that all the PRG computations are correct) and the randomness $r_{i,h}$ that party P_i used in the protocol Π_h from $\{y_{i,j}\}_{j \in H}$ (output by $\mathsf{Sim}_{\mathsf{WL}}$). We give $\{\phi_1^{i \to h}\}_{i \in M, h \notin C \cup K}$ as the first round message sent by the corrupted clients to the honest servers.

7. Sim_Φ queries its own ideal functionality on input $\{z_i\}_{i \in M}$. We use this to compute $(y, \sigma_1, \ldots, \sigma_n)$ honestly (using the private inputs of the honest parties) and provide this as the output from the ideal functionality to Sim_Φ.
8. Sim_Φ outputs the second round messages $\{\phi_2^h\}_{h \notin C \cup K}$ as the second round message sent by the honest servers. For each $h \notin C \cup K$, we provide ϕ_2^h as the output of the functionality along with the consistent input, randomness pair of the corrupted parties to Sim_{Π_h} and obtain the final round message from Sim_{Π_h} on behalf of all unaborted honest parties. We use this to generate the view of $\mathcal{A}$ as well as compute the output of all the honest parties as in the previous hybrid.

In Lemma 9.5, we show that $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$ from the security of the outer protocol against adversaries running in time $T_1 \cdot \mathsf{poly}(\lambda)$.

- $\underline{\mathsf{Hyb}_5}$: In this hybrid, we make the following two changes:
  - When Sim_Φ queries the ideal functionality on $\{z_i = (x_i, k_i)\}_{i \in M}$, we query f on $\{x_i\}_{i \in M}$ and obtain the output y. For each $i \in M$, we compute $\sigma_i := \mathsf{MAC}(k_i, y)$ and for each $i \in H$, we choose σ_i uniformly at random.
  - In the output phase, we recover $(y', \sigma_1', \ldots, \sigma_n')$ as in the previous hybrid and then check if $y' = y$ and if for each $i \in H$, if $\sigma_i' = \sigma_i$. For every $i \in H$, such that above check passes, we instruct the ideal functionality to deliver the outputs to P_i. For all other parties, we instruct them to abort.

  Hyb_5 is statistically close to Hyb_4 from the one-time security of the MAC scheme and the uniformity of the tags property of the one-time MAC scheme.

- $\underline{\mathsf{Hyb}_6}$: In this hybrid, we generate the first round messages that the honest clients send to corrupt servers $C \cup K$ as $\Phi_1(1^\lambda, i, \mathbf{0})$ where $\mathbf{0}$ is a default input.

  Hyb_6 is identically distributed to Hyb_5 from the property of the outer protocol that the distribution of the first round messages from the honest clients to the corrupted servers that is generated by Sim_Φ is identically distributed to the messages generated honestly using a default input.

- $\underline{\mathsf{Hyb}_7}$: In this hybrid, we make the following changes:
  1. We fix the first round message in the protocol.
  2. We run the adversary $\mathcal{A}$ in the first three rounds as in the previous hybrid. If $\mathcal{A}$ aborts in this phase or sends an invalid message, then we abort the execution and output the view of $\mathcal{A}$ and instruct all the honest parties to output $\bot$. We call this execution as the first rewind thread.
  3. If the adversary does not abort nor sends an invalid message in the rewind thread, then we estimate the probability that adversary sends such a valid message. Specifically, we repeatedly run $\mathcal{A}$ (by sampling independent second round and third round messages) until we obtain 12λ executions where the adversary's third round message is valid. Let T be the total number of trials until we obtain 12λ successful executions. We set $\widetilde{\epsilon} = 12\lambda/T$.

4. We then go back to the main thread and repeat the following for $\lambda^2/\widetilde{\epsilon}$ times. In each trial, we generate an independent second round and third round message and wait until the adversary does not abort and does not send any invalid messages. If the adversary fails to send a valid third round message in each of the trials, we output the special symbol ABORT. Otherwise, we continue the execution in the main thread as before and output the view of the adversary in the main thread and compute the outputs of all the honest parties in the main thread.

In Lemma 9.6, we show that $\mathsf{Hyb}_6 \approx_s \mathsf{Hyb}_7$.

- $\underline{\mathsf{Hyb}_8}$: In this hybrid, we make the following changes:
  1. In the first rewind thread, we additionally run $\mathsf{SPExt}_{\mathsf{WL},R}$ (running in time T_2) to compute $\{x'_{i,j}\}_{i\in M,j\in H}$. We parse each $x'_{i,j}$ as a set of $[m]$ of size λ and we set $K' = \cup_{i\in M,j\in H} x'_{i,j}$.
  2. In the main thread, we use this value K' instead of K in the main thread as the set of inner protocol executions that are generated honestly and the set of servers that are corrupted initially in the outer protocol. Note that to compute the output of the watchlist protocol, we still use $\{x_{i,j}\}_{i\in M,j\in H}$ that is output by $\mathsf{Sim}_{\mathsf{WL}}$.

  By the property of $\mathsf{SPExt}_{\mathsf{WL},R}$ it follows that K' computed above is a superset of K and $|K'| \leq \lambda n^2$. Thus, via identical arguments given in Lemma 9.4 and Lemma 9.2, we can use the security of the outer and the inner protocols respectively against adversaries running in time $(T_1 + T_2)\cdot \mathsf{poly}(\lambda)$ to corrupt the set of servers corresponding to K' instead of K.

- $\underline{\mathsf{Hyb}_9}$: In this hybrid, we make the following changes:
  1. In the first rewind thread, we run $\mathsf{Sim}_{\mathsf{WL}}$ on the first round message generated by $\mathcal{A}$ to compute the set K as in the previous hybrid.
  2. We run $\phi_1(1^\lambda, i, \mathbf{0})$ to compute $\{\overline{\phi}_1^{i\to h}\}_{h\in[m]}$ for each $i \in H$.
  3. For each $h \notin K$ in the first rewind thread as well as the rewind threads used in computing $\widetilde{\epsilon}$, we generate the first two round messages on behalf of the honest parties for the protocol Π_h using the input $\{\overline{\phi}_1^{i\to h}\}_{i\in H}$ and uniformly chosen randomness $r_{i,h}$. These messages were generated using the simulator Sim_{Π_h} in the previous hybrid.

  Via an identical argument to Lemma 9.2, in each of the rewind threads (including the ones used in estimating $\widetilde{\epsilon}$), we can rely on the security of the inner protocol against adversaries running in time $(T_1 + T_2)\cdot \mathsf{poly}(\lambda)$ to show that $\mathsf{Hyb}_8 \approx_c \mathsf{Hyb}_9$.

- $\underline{\mathsf{Hyb}_{10}}$: In this hybrid, we make the following changes:
  1. In both the main thread and in the first rewind thread, we run the watchlist protocol WL using honestly chosen $x'_{i,j}$ and $y_{i,j} = \{r_{i,h}, \overline{\phi}_1^{i\to h}\}_{h\in[m]}$ (where $r_{i,h}$ is the same as the one used in the main thread for each $h \in K'$) for each $i \in H$ and $j \in M$. Note that in the previous hybrid, we generated the messages of the watchlist protocol using $\mathsf{Sim}_{\mathsf{WL}}$.
  2. We run $\mathsf{Ext}_{\mathsf{WL},S}$ on the rewind thread to extract the inputs $\{y_{i,j}\}_{i\in M,j\in H}$ and use this to continue with the rest of the execution in the main thread as in the previous hybrid.

In Lemma 9.7, we show that Hyb_9 is computationally indistinguishable to Hyb_{10} from the 1-rewinding sender non-malleability property of the watchlist protocol.

- $\underline{\mathsf{Hyb}_{11}}$: In this hybrid, we use the rewinding extractor $\mathsf{Ext}_{\mathsf{WL},R}$ to compute $\{x'_{i,j}\}_{i \in M, j \in H}$ instead of running the straightline extractor $\mathsf{SPSExt}_{\mathsf{WL},R}$. This hybrid is statistically close to the previous hybrid since the output of these two extractors are the same except with negligible probability. We note that Hyb_{11} identically distributed to the ideal execution with ideal world adversary Sim.

Lemma 9.2. *Assuming the super-polynomial time simulation security of the watchlist protocol* WL*, we have* $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$.

Proof. Assume for the sake of contradiction that Hyb_0 and Hyb_1 are computationally distinguishable. We now show that this contradicts the super-polynomial time simulation security of the WL.

We start interacting with the watchlist external challenger by providing the inputs $\{x_{i,j}, y_{i,j}\}_{i \in H, j \in [n] \setminus \{i\}}$ of the honest parties. Let $\mathcal{A}_1$ be the adversary defined in the description of Hyb_1. We provide the external challenger with $\mathcal{A}_1$ as the adversary against the watchlist protocol. The external challenger provides the view of $\mathcal{A}_1$ and the output of all the honest parties in the watchlist protocol. We extract the view of $\mathcal{A}$ in the first three rounds from the view of $\mathcal{A}_1$ and we use the output of the honest parties in the watchlist protocol to perform the same checks as described in Figure 12 at the end of the third round. We generate the final round message and compute the view of $\mathcal{A}$ in the overall protocol along with the outputs of all the honest parties (as described in Hyb_0). We run the distinguisher between Hyb_0 and Hyb_1 on this and output whatever the distinguisher outputs.

If the view of $\mathcal{A}_1$ and the output of the honest parties in the watchlist protocol are generated by the external challenger as in the real execution, then the input to the distinguisher is distributed identically to Hyb_0. Otherwise, the input to the distinguisher is identical to Hyb_1. Thus, if the distinguisher can distinguish between Hyb_0 and Hyb_1 with non-negligible advantage, the above reduction breaks the super-polynomial time simulation security of the watchlist protocol and this is a contradiction. □

Lemma 9.3. *Assuming the security of the inner protocol against adversaries running in time* $T_1 \cdot \mathsf{poly}(\lambda)$*, we have* $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$.

Proof. Assume for the sake of contradiction that there exists a distinguisher that can distinguish between Hyb_1 and Hyb_2 with non-negligible advantage. We will use this distinguisher to break the security of the inner protocol.

We define the adversary $\mathcal{A}_2$ as described in Hyb_2. This adversary interacts with the simulator $\mathsf{Sim}_{\mathsf{WL}}$ for the watchlist protocol and generates the messages for the inner protocol executions by interacting with the external challenger. $\mathsf{Sim}_{\mathsf{WL}}$ provides the first round message of the watchlist protocol to $\mathcal{A}_2$ and it forwards it to $\mathcal{A}$. $\mathcal{A}_2$ receives the first round message from $\mathcal{A}$ corresponding to the watchlist protocol and forwards it to $\mathsf{Sim}_{\mathsf{WL}}$. $\mathsf{Sim}_{\mathsf{WL}}$ runs in super-polynomial (specifically, time T_1) time and computes $\{x_{i,j}\}_{i \in M, j \in H}$ from this message. We interpret $x_{i,j}$ as a subset of $[m]$ of size λ and denote $K = \cup_{i \in M, j \in H} x_{i,j}$.

$\mathcal{A}_2$ begins interacting with the external challenger for Π_h to generate messages in executions $h \notin K$. $\mathcal{A}_2$ provides the external challenger with the honest party inputs $\{\phi_1^{i \to h}\}_{i \in H}$ in Π_h for each $h \notin K$. For each $h \in K$, $\mathcal{A}_2$ generates the messages in the protocol Π_h using honest party inputs $\{\phi_1^{i \to h}\}_{i \in H}$ and uniformly chosen randomness $\{r_{i,h}\}_{i \in H}$. We run $\mathsf{Sim}_{\mathsf{WL}}$ on $\mathcal{A}_2$ and obtain the view of $\mathcal{A}_2$, $\{y_{i,j}\}_{i \in M, j \in H}$ and instruction for each $i \in H$ to either obtain the output of the watchlist functionality to P_i or obtain $\bot$. We extract the view of $\mathcal{A}$ in the first three rounds from $\mathsf{View}_{\mathcal{A}_2}$.

For each $i \in M$ and $j \in H$, we parse $y_{i,j}$ as $(r_{i,h}, \phi_1^{i \to h})$ and check if the messages in the protocol Π_h sent by P_i are consistent with this randomness, input pair. If for each $j \in H$, this pair is inconsistent, or if the PRG computations in $\phi_1^{i \to h}$ are incorrect, then we provide an inconsistent input, randomness pair on behalf of the corrupted parties to the external challenger. If for each $i \in M$, there exists at least one $j \in H$ such that the input randomness pair $(r_{i,h}, \phi_1^{i \to h})$ in $y_{i,j}$ is consistent with all the messages sent in Π_h and if the PRG computations in $\phi_1^{i \to h}$ are correct, then we forward this to the external challenger. In either case, we obtain the final round message from the external challenger and we use this to generate the view of $\mathcal{A}$ in the entire protocol as well as compute the output of all the honest parties as in the previous hybrid. Finally, we run the distinguisher between Hyb_1 and Hyb_2 on the view of $\mathcal{A}$ and the outputs of the honest parties and output whatever the distinguisher outputs.

We note that if the messages in the inner protocol Π_h for each $h \notin K$ is generated by the external challenger using the honest party's inputs and uniform randomness, then the input to the distinguisher is identically distributed to Hyb_1. Else, it is identically distributed to Hyb_2. The running time of the above reduction is bounded by $T_1 \cdot \mathsf{poly}(\lambda)$ as the running time of $\mathsf{Sim}_{\mathsf{WL}}$ is upper bounded by T_1. Since the distinguisher is assumed to distinguisher between Hyb_1 and Hyb_2 with non-negligible advantage, this contradicts the security of the inner protocol against adversaries running in time $T_1 \cdot \mathsf{poly}(\lambda)$. □

Lemma 9.4. $\mathsf{Hyb}_2 \approx_s \mathsf{Hyb}_3$.

Proof. We show that if $|C| > \lambda \cdot n^2$ then every honest party aborts at the end of the third round in Hyb_2.

Let us fix some honest party $i \in H$. Note that its watched executions K_i is uniformly distributed and is independent of the view of the adversary in the first three rounds. This follows since we are running $\mathsf{Sim}_{\mathsf{WL}}$ in Hyb_2 to generate the messages of the honest parties in the watchlist protocol. We now show that if $|C| > \lambda \cdot n^2$ then the probability that $|K_i \cap C| = 0$ is at most $2^{-O(\lambda)}$. If $|K_i \cap C| \neq 0$ then the honest party i aborts at the end of the third round.

$$\begin{aligned}
\Pr[|K_i \cap C| = 0] &= \frac{\binom{m-|C|}{\lambda}}{\binom{m}{\lambda}} \\
&\leq \frac{\binom{m-\lambda \cdot n^2}{\lambda}}{\binom{m}{\lambda}} \\
&= (1 - \frac{\lambda \cdot n^2}{m})(1 - \frac{\lambda \cdot n^2}{m-1}) \ldots (1 - \frac{\lambda \cdot n^2}{m-(\lambda-1)}) \\
&< (1 - \frac{\lambda \cdot n^2}{m})^{\lambda} \\
&\leq e^{-O(\lambda)} \quad \text{(Since } m = O(\lambda \cdot n^2))
\end{aligned}$$

By an union bound, the probability that there exists an honest party that does not abort is at most $n \cdot e^{-O(\lambda)} = 2^{-O(\lambda)}$. □

Lemma 9.5. *Assuming the privacy with knowledge of outputs property of the outer protocol against adversaries running in time* $T_1 \cdot \mathsf{poly}(\lambda)$*, we have* $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$.

Proof. Assume for the sake of contradiction that Hyb_3 and Hyb_4 are distinguishable with non-negligible advantage. We will show that this contradicts the security of the outer protocol against adversaries running in time $T_1 \cdot \mathsf{poly}(\lambda)$.

We define the adversary $\mathcal{A}_3$ as described in Hyb_4. We compute the set K as described there. We start interacting with the challenger for the outer protocol by corrupting the set of clients indexed by M and the set of servers indexed by K. We provide $\{z_i\}_{i \in H}$ to the external challenger as the honest clients inputs and obtain $\{\phi_1^{i \to h}\}_{h \in K}$ as the first round message from the honest clients to the corrupted servers. We use this to complete the execution of the first three rounds with $\mathcal{A}$ as given in the hybrid description. $\mathsf{Sim}_{\mathsf{WL}}$ at the end of the third round outputs the view of $\mathcal{A}_3$, $\{y_{i,j}\}_{i \in M, j \in H}$ along with instruction for each $i \in H$ to either obtain the output from the watchlist or obtain abort. We extract the view of $\mathcal{A}$ in the first three rounds from the view of $\mathcal{A}_3$. We construct the set C as described in the hybrid and abort if $|C| > \lambda \cdot n^2$. Otherwise, we instruct the challenger to adaptively corrupt the set of servers indexed by C and obtain the first round messages $\{\phi_1^{i \to h}\}_{i \in H, h \in C}$ from the challenger. We provide this to Sim_{Π_h} for each $h \in C$ as the inputs of the honest parties. For each $h \notin C \cup K$ and for each $i \in M$, we first extract the consistent input $\phi_1^{i \to h}$ (such that the PRG computations are all correct) and the randomness $r_{i,h}$ that party P_i in the protocol Π_h from $\{y_{i,j}\}_{j \in H}$ (output by $\mathsf{Sim}_{\mathsf{WL}}$). We give $\{\phi_1^{i \to h}\}_{i \in M, h \notin C \cup K}$ to the external challenger as the first round message sent by the corrupted clients to the honest servers. The challenger outputs the second round messages $\{\phi_2^h\}_{h \notin C \cup K}$ sent by the honest servers. For each $h \notin C \cup K$, we provide ϕ_2^h as the output of the functionality along with the consistent input, randomness pair of the corrupted parties to Sim_{Π_h} and obtain the final round message from Sim_{Π_h} on behalf of all unaborted honest parties. We use this to generate the view of $\mathcal{A}$ as well as compute the output of all the honest parties as in the previous hybrid. We finally run the distinguisher between Hyb_3 and Hyb_4 on this and output whatever the distinguisher outputs.

We note that $|K \cup C| \leq 2\lambda n^2$ and hence, the above reduction emulates a valid adversary against the outer protocol. Further, the running time of the adversary is upper bounded by $T_1 \cdot \mathsf{poly}(\lambda)$ since the running time of $\mathsf{Sim}_{\mathsf{WL}}$ is bounded by T_1. We note that if the messages for the outer protocol are generated by the challenger as in the real execution then the input to the distinguisher is identical to Hyb_3 and is otherwise, identically distributed to Hyb_4. Since we assumed that the distinguisher can distinguish between Hyb_3 and Hyb_4 with non-negligible advantage, this contradicts the security of the outer protocol against adversaries running in time $T_1 \cdot \mathsf{poly}(\lambda)$. □

Lemma 9.6. $\mathsf{Hyb}_6 \approx_s \mathsf{Hyb}_7$

Proof. The only difference between Hyb_6 and Hyb_7 is that Hyb_7 sometimes outputs the special symbol ABORT whereas Hyb_6 never outputs this. We now argue that the probability that Hyb_7 outputs the special symbol ABORT is negligible.

By a standard argument given in [GK96a], we note that $\widetilde{\epsilon}$ is within a constant factor of ϵ except with probability $2^{-\lambda}$. Thus, the probability that in each of $\frac{\lambda^2}{\widetilde{\epsilon}}$ trials that the adversary aborts or fails to send a valid message is at most $(1-\epsilon)^{\frac{\lambda^2}{\widetilde{\epsilon}}} < e^{-O(\lambda)}$ if $\widetilde{\epsilon}$ is within a constant factor of ϵ. Thus, Hyb_6 and Hyb_7 are statistically close. □

Lemma 9.7. *Assuming the 1-rewinding non-malleable security of the watchlist protocol against adversaries running in time* $T_2 \cdot \mathsf{poly}(\lambda)$*, we have* $\mathsf{Hyb}_9 \approx_c \mathsf{Hyb}_{10}$.

Proof. Assume for the sake of contradiction that Hyb_8 and Hyb_9 are computationally distinguishable with non-negligible advantage. We use this to break the 1-rewinding security of the watchlist protocol against adversaries running in time $T_2 \cdot \mathsf{poly}(\lambda)$.

We start interacting with a watchlist challenger and provide honestly chosen $x'_{i,j}$ for each $i \in H$ and $j \in M$ and $y_{i,j} = \{r_{i,h}, \overline{\phi}_1^{i \to h}\}_{h \in [m]}$ as the honest party's inputs. We define an adversary $\mathcal{A}_4$ that completes the execution of the first rewind thread as in Hyb_9 but obtains the messages in the watchlist protocol externally. We provide $\mathcal{A}_4$ to the external challenger and the challenger outputs the view of $\mathcal{A}_4$ along with $\{y_{i,j}\}_{i \in M, j \in H}$. We extract the view of the $\mathcal{A}$ in the three rounds of the first rewind thread from the view of $\mathcal{A}_4$. If $\mathcal{A}$ has aborted or sent an invalid message, then we run the distinguisher between Hyb_9 and Hyb_{10} on the view of $\mathcal{A}$ generated above and output of all the honest parties set to $\perp$. We output whatever this distinguisher outputs. Else, we first compute K' using the super-polynomial time extractors $\mathsf{SPExt}_{\mathsf{WL},R}$ (running in time T_2) on the view of $\mathcal{A}$ in the first rewind thread. We define a new adversary $\mathcal{A}_5$ that has the first round messages of the watchlist protocol to be the same as that of $\mathcal{A}_4$ but completes the execution of the second and the third round of the protocol as in the main thread of Hyb_9. We provide the challenger with $\mathcal{A}_5$ and it provides with the view of $\mathcal{A}_5$. We extract the view of $\mathcal{A}$ in the first three rounds of the main thread from view of $\mathcal{A}_5$. If $\mathcal{A}$ had aborted or sent an invalid message in this view, then we output a random bit to the external challenger and abort the execution. Otherwise, we use the view of $\mathcal{A}$ in the first three rounds along with $\{y_{i,j}\}_{i \in M, j \in H}$ output by the external challenger and K' computed as above to complete the execution with the adversary and generate the view of the adversary in the main thread along with the output of the honest parties as in Hyb_9. We finally run the distinguisher between Hyb_9 and Hyb_{10} on this and output whatever it outputs.

We note that the above reduction runs in time $T_2 \cdot \mathsf{poly}(\lambda)$. We note that if the messages in the watchlist protocol are generated by the challenger using the real inputs then the input to the distinguisher is identical to Hyb_{10}. Otherwise, it is distributed identically to Hyb_9. Let the probability that the distinguisher correctly predicts whether it is given a sample from Hyb_9 or Hyb_{10} be $1/2 + 1/q(\lambda)$ for some polynomial $q(\cdot)$. Let ϵ be the probability that the adversary does not abort nor send an invalid message in the first rewind thread. From the previous hybrid arguments, it follows that the probability that the adversary does the same in the main thread is at most $\epsilon + \mathsf{negl}(\lambda)$ (and is lower bounded by $\epsilon - \mathsf{negl}(\lambda)$). We now estimate the probability that this reduction breaks the 1-rewinding non-malleable security of the watchlist protocol. This probability is

$$\begin{aligned}
&\geq \quad (1-\epsilon)(1/2 + 1/q(\lambda)) + \epsilon(\epsilon - \mathsf{negl}(\lambda))(1/2 + 1/q(\lambda)) + \epsilon(1 - \epsilon - \mathsf{negl}(\lambda))1/2 \\
&\geq \quad 1/2 + (1 + \epsilon^2 - \epsilon)(1/q(\lambda)) - \mathsf{negl}(\lambda) \\
&\geq \quad 1/2 + (3/4)(1/q(\lambda)) - \mathsf{negl}(\lambda)
\end{aligned}$$

Thus, the above reduction breaks the 1-rewinding non-malleable security of the watchlist protocol and this is a contradiction. □

Acknowledgments. Y. Ishai was supported in part by ERC Project NTSC (742754), BSF grant 2018393, ISF grant 2774/20, and a Google Faculty Research Award. D. Khurana was supported in part by NSF CAREER CNS-2238718 and DARPA SIEVE. A. Sahai was supported in part from a Simons Investigator Award, DARPA SIEVE award, NTT Research, NSF Frontier Award 1413955, BSF grant 2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, and an

Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through Award HR00112020024. A. Srinivasan was supported in part by a SERB startup grant and Google India Research Award.

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.

[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.

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

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

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

[AJL$^+$12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *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.

[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.

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

[BGI$^+$17] Saikrishna Badrinarayanan, Sanjam Garg, Yuval Ishai, Amit Sahai, and Akshay Wadia. Two-message witness indistinguishability and secure computation in the plain model from new assumptions. In *ASIACRYPT*, 2017.

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

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

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

[BS19] Saikrishna Badrinarayanan and Akshayaram Srinivasan. Revisiting non-malleable secret sharing. LNCS, pages 593–622. Springer, Heidelberg, Germany, 2019.

[CCG+20] Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. On round optimal secure multiparty computation from minimal assumptions. *To appear in TCC*, 2019:216, 2020.

[CCG+21] Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Oblivious transfer from trapdoor permutations in minimal rounds. In *TCC 2021, Part II*, pages 518–549, 2021.

[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.

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

[DGI+19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. LNCS, pages 3–32, Santa Barbara, CA, USA, 2019. Springer, Heidelberg, Germany.

[FK84] Michael L Fredman and János Komlós. On the size of separating systems and families of perfect hash functions. *SIAM Journal on Algebraic Discrete Methods*, 5(1):61–68, 1984.

[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.

[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 *TCC 2018, Part I*, LNCS, pages 123–151. Springer, Heidelberg, Germany, March 2018.

[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.

[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.

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

[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 Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 1128–1141, Cambridge, MA, USA, June 18–21, 2016. ACM 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.

[GSZ21] Vipul Goyal, Akshayaram Srinivasan, and Chenzhi Zhu. Multi-source non-malleable extractors and applications. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 468–497. Springer, 2021.

[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.

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

[HPV20] Carmit Hazay, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. Which languages have 4-round fully black-box zero-knowledge arguments from one-way functions? In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT*

2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III, volume 12107 of *Lecture Notes in Computer Science*, pages 599–619. Springer, 2020.

[HV18] Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. Round-optimal fully black-box zero-knowledge arguments from one-way permutations. In *TCC 2018, Part I*, LNCS, pages 263–285. Springer, Heidelberg, Germany, March 2018.

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

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

[IKSS21] Yuval Ishai, Dakshita Khurana, Amit Sahai, and Akshayaram Srinivasan. On the round complexity of black-box secure MPC. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 214–243. Springer, 2021.

[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.

[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.

[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.

[KOS18] Dakshita Khurana, Rafail Ostrovsky, and Akshayaram Srinivasan. Round optimal black-box "commit-and-prove". In *TCC 2018, Part I*, LNCS, pages 286–313. Springer, Heidelberg, Germany, March 2018.

[KS17] Dakshita Khurana and Amit Sahai. Two-message non-malleable commitments from standard sub-exponential assumptions. *IACR Cryptology ePrint Archive*, 2017:291, 2017.

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

[MOSV22] Varun Madathil, Chris Orsini, Alessandra Scafuro, and Daniele Venturi. From privacy-only to simulatable OT: black-box, round-optimal, information-theoretic. In *ITC 2022, volume = 230, pages = 5:1–5:20,*, 2022.

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

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

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

[PS21] Arpita Patra and Akshayaram Srinivasan. Three-round secure multiparty computation from black-box two-round oblivious transfer. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 185–213. Springer, 2021.

[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 Extractable Commitment with Straight-Line Extraction

In this section, we give a construction of a three-round commitment scheme that has straight-line extraction by a super-polynomial time extractor.

A.1 Definition

Definition A.1. *A three-round extractable commitment scheme* $(\mathsf{ECom}_1, \mathsf{ECom}_2, \mathsf{ECom}_3, \mathsf{Verify})$ *is said to admit a super-polynomial time, straight-line extraction if it satisfies the following properties:*

- ***Completeness:*** *For every input message* msg *of the committer* C*, the output of* Verify *on a transcript generated by the interaction* $\langle R(1^\lambda), C(1^\lambda, \mathsf{msg})\rangle$ *and the internal randomness of the receiver outputs 1 with probability 1.*
- ***Hiding against Malicious Receivers.*** *For every adversary* $\mathcal{A}$ *corrupting the receiver* R *and for any two input messages* $\mathsf{msg}_0, \mathsf{msg}_1$ *of the committer* C*:*

$$\left\{\mathsf{View}_{\mathcal{A}}(\langle \mathcal{A}(1^\lambda), C(1^\lambda, \mathsf{msg}_0)\rangle)\right\} \approx_c \left\{\mathsf{View}_{\mathcal{A}}(\langle \mathcal{A}(1^\lambda), C(1^\lambda, \mathsf{msg}_1)\rangle)\right\}$$

- ***Extraction:*** *For every adversary $\mathcal{A}$ that corrupts the committer, there exists an expected PPT machine* Ext *and a super-polynomial time, straight-line machine* SPExt *that make black-box use of $\mathcal{A}$ and output* $(\mathsf{View}_{\mathcal{A}}, \mathsf{msg})$ *such that:*

  1. $\mathsf{View}_{\mathcal{A}}$ *output by* Ext *is identically distributed to* $\mathsf{View}_{\mathcal{A}}(\langle R(1^\lambda), \mathcal{A}(1^\lambda)\rangle)$.
  2. *If there exists some message* msg' *and random tape r such that the committer transcript in* $\mathsf{View}_{\mathcal{A}}$ *is consistent with input* msg' *using the random tape r, then* $\mathsf{msg}' = \mathsf{msg}$ *except with negligible probability where* $(\mathsf{View}_{\mathcal{A}}, \mathsf{msg}) \leftarrow (\mathsf{Ext})^{\mathcal{A}}(1^\lambda)$.
  3. $(\mathsf{SPExt})^{\mathcal{A}}(1^\lambda) \approx_s (\mathsf{Ext})^{\mathcal{A}}(1^\lambda)$.

A.2 Building Blocks

The construction uses the following building blocks.

Extractable Commitment. A three round extractable commitment scheme $(\mathsf{ECom}_1, \mathsf{ECom}_2, \mathsf{ECom}_3)$ that is (T_2, ϵ)-hiding against malicious receivers running in time T_2 and satisfies over extraction (see Definition 3.5).

Pairwise Verifiable Secret Sharing. A pairwise verifiable, k-out-of-m secret sharing scheme $(\mathsf{Share}, \mathsf{Rec})$ that has the following properties:

- **Correctness.** For any secret s,

$$\Pr[\mathsf{Rec}(\mathsf{Sh}_1, \ldots, \mathsf{Sh}_m) = s : (\mathsf{Sh}_1, \ldots, \mathsf{Sh}_m) \leftarrow \mathsf{Share}(s)] = 1$$

- **Perfect Secrecy.** For any two secrets s_0, s_1 and for any subset $K \subset [m]$ of size k, we have:

$$\left\{\mathsf{Sh}_K : (\mathsf{Sh}_1, \ldots, \mathsf{Sh}_m) \leftarrow \mathsf{Share}(s_0)\right\} \equiv \left\{\mathsf{Sh}_K : (\mathsf{Sh}_1, \ldots, \mathsf{Sh}_m) \leftarrow \mathsf{Share}(s_1)\right\}$$

- **Pairwise Verifiability.** If there exists a set $K \subseteq [m]$ of size at least $S(m,k)$ such that for every $i, j \in K$, $(\mathsf{Sh}_i, \mathsf{Sh}_j)$ are pairwise consistent, then for any value of $\mathsf{Sh}_{[m]\setminus K}$, the output of $\mathsf{Rec}(\mathsf{Sh}_1, \ldots, \mathsf{Sh}_m)$ is the same.

We set $k = 2\lambda$, $m = c\lambda$ for some large constant c, and $S(m,k) = m - k$.

Setting the Parameters. We set $T_2 \geq \mathsf{poly}(\binom{m}{k} \cdot \lambda)$ and $\epsilon \leq \mathsf{negl}(\binom{m}{k} \cdot \lambda)$.

A.3 Construction

We describe the construction in Figure 13.

- **Round-1:** C does the following:
  1. It computes $(\mathsf{Sh}_1, \dots, \mathsf{Sh}_m) \leftarrow \mathsf{Share}(\mathsf{msg})$.
  2. For each $i \in [m]$,
     (a) It chooses $r_i \leftarrow \{0,1\}^*$ as the randomness for an extractable commitment scheme.
     (b) It computes $\mathsf{Com}_1^i := \mathsf{ECom}_1(\mathsf{Sh}_i; r_i)$.
  3. It sends $\{\mathsf{Com}_1^i\}_{i \in [m]}$ to R.
- **Round-2:** R does the following:
  1. It chooses a random set $K \subset [m]$ of size λ.
  2. For each $i \in [m]$,
     (a) It computes $\mathsf{Com}_2^i \leftarrow \mathsf{ECom}_2(\mathsf{Com}_1^i)$.
  3. It sends $\{\mathsf{Com}_2^i\}_{i \in [m]}, K$.
- **Round-3:** C does the following:
  1. For each $i \in [m]$,
     (a) It computes $\mathsf{Com}_3^i := \mathsf{ECom}_3(\mathsf{Com}_2^i, \mathsf{Sh}_i; r_i)$.
  2. It sends $\{\mathsf{Com}_3^i\}_{i \in [m]}, \{\mathsf{Sh}_i, r_i\}_{i \in K}$ to R.
- **Verification:** R does the following:
  1. For each $i \in K$,
     (a) It checks if $\{\mathsf{Com}_1^i, \mathsf{Com}_3^i\}$ is correctly computed extractable commitment to Sh_i using randomness r_i.
     (b) If it is not the case, then it outputs 0.
  2. For each $i, j \in K$,
     (a) It checks if Sh_i and Sh_j are pairwise consistent.
     (b) If it is not the case, it outputs 0.
  3. Else, it runs $\mathsf{Verify}_{\mathsf{ECom}}$ on $\{\mathsf{Com}_1^i, \mathsf{Com}_2^i, \mathsf{Com}_3^i\}$ and its internal randomness for each $i \in [m]$ and if any of the checks fail, it outputs 0. If all the checks pass, it outputs 1.

Figure 13: Three-Round Extractable Commitment satisfying Definition A.1

A.4 Proof of Security

A.4.1 Hiding against Malicious Receivers

We show this via a hybrid argument.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to $\mathsf{View}_{\mathcal{A}}(\langle \mathcal{A}(1^\lambda), C(1^\lambda, \mathsf{msg}_0) \rangle)$.
- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we make the following changes:
  1. We repeat the following for $\lambda \cdot \binom{m}{\lambda}$ iterations.
     (a) Before sending the first round message, we randomly choose a subset $K' \subset [m]$ of size λ.
     (b) After receiving the second round message from $\mathcal{A}$, we check if $K = K'$, we proceed to complete the execution with $\mathcal{A}$. Else, we move to the next iteration.

2. If fail to proceed in each of $\lambda \cdot \binom{m}{\lambda}$ iterations, we output a special symbol fail.

By an identical argument to Lemma 4.11, we can show that $\mathsf{Hyb}_0 \approx_s \mathsf{Hyb}_1$.

- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we make the following changes to each of the $\lambda \cdot \binom{m}{\lambda}$ iterations:

  1. For each $i \notin K'$, we generate $\{\mathsf{Com}_1^i, \mathsf{Com}_3^i\}$ as extractable commitments to some dummy message instead of commitment to Sh_i.

  By an identical argument to Lemma 4.12, we can rely on the (T_2, ϵ)-hiding property of ECom to show that $\mathsf{Hyb}_1 \approx_{(T_2,\epsilon)} \mathsf{Hyb}_2$.

- $\underline{\mathsf{Hyb}_3}$: In this hybrid, in each of the $\lambda \cdot \binom{m}{\lambda}$ iterations, we generate $\{\mathsf{Sh}_i\}_{i\in K'}$ using $\mathsf{Share}(\mathsf{msg}_1)$ instead of $\mathsf{Share}(\mathsf{msg}_0)$. It follows from the perfect secrecy of the secret sharing scheme that Hyb_3 and Hyb_2 are identically distributed.

- $\underline{\mathsf{Hyb}_4}$: In this hybrid, we compute $(\mathsf{Sh}_1, \ldots, \mathsf{Sh}_m) \leftarrow \mathsf{Share}(\mathsf{msg}_1)$ and make the following changes to each of the $\lambda \cdot \binom{m}{\lambda}$ iterations:

  1. For each $i \notin K'$, we generate $\{\mathsf{Com}_1^i, \mathsf{Com}_3^i\}$ as extractable commitments to Sh_i instead of commitment to a dummy message.

  This just corresponds to reversing the changes made in Hyb_2 and hence, by the exact same argument, we have $\mathsf{Hyb}_3 \approx_{(T_2,\epsilon)} \mathsf{Hyb}_4$.

- $\underline{\mathsf{Hyb}_5}$: In this hybrid, we reverse the changes made in Hyb_1. Via an identical argument given above, we can show that Hyb_5 and Hyb_4 are statistically close. We note that Hyb_5 is distributed identically to $\mathsf{View}_{\mathcal{A}}(\langle \mathcal{A}(1^\lambda), C(1^\lambda, \mathsf{msg}_1)\rangle)$.

A.4.2 Extraction

We begin with the description of Ext. Ext runs the honest receiver strategy and outputs the view of $\mathcal{A}$ and $\mathsf{msg} = \bot$ if any one of the checks made by the honest receiver fails. If all the checks pass, it runs the extractor for ECom on the adversary and obtains for each $i \in [m]$, the committed value Sh_i. It computes $\mathsf{msg} := \mathsf{Rec}(\mathsf{Sh}_1, \ldots, \mathsf{Sh}_m)$. Finally, it outputs the view of $\mathcal{A}$ and msg. It is straightforward to see that the view of the adversary generated by Ext is identically distributed to its view when interacting with an honest receiver. Additionally, it follows from the property of the rewinding extractor for ECom that if the commitments $\{\mathsf{Com}_1^i, \mathsf{Com}_3^i\}_{i\text{ß}[m]}$ are generated correctly, then $\{\mathsf{Sh}_i\}_{i\in[m]}$ output by this extractor is the value committed except with negligible probability. This is sufficient to show the second property under extraction given in Definition A.1.

We now give the description of SPExt. SPExt is exactly same as Ext except that it obtains $\{\mathsf{Sh}_i\}_{i\in[m]}$ by running the super-polynomial time, straight-line extractor for ECom. It then computes msg exactly as described in Ext and finally outputs the view of adversary along with msg. We now argue that the output of SPExt and Ext are statistically indistinguishable.

Lemma A.2. $(\mathsf{SPExt})^{\mathcal{A}}(1^\lambda) \approx_s (\mathsf{Ext})^{\mathcal{A}}(1^\lambda)$.

Proof. We show this through a sequence of hybrids.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the output of $(\mathsf{SPExt})^{\mathcal{A}}(1^\lambda)$.

- $\underline{\mathsf{Hyb}_1}$: In this hybrid, we make the following changes:
  1. We non-uniformly fix the first round message from the committer and extract the values inside $\{\mathsf{Com}_1^i\}_{i\in[m]}$.
  2. We initialize an empty set C and for each $i \in [m]$, it adds i to C if the Com_1^i is generated incorrectly.
  3. If $|C| > \lambda$, then we set $\mathsf{msg} = \bot$.
  4. Else, we compute msg as before and output the view of $\mathcal{A}$ and msg.

  We note that if $|C| > \lambda$, the probability that $|C \cap K| = 0$ is $2^{-O(\lambda)}$ (since K is uniformly chosen random subset of $[m]$; see Claim 4.4). Thus, except with negligible probability the output of Hyb_0 and Hyb_1 are exactly the same.

- $\underline{\mathsf{Hyb}_2}$: In this hybrid, we make the following changes:
  1. We initialize an empty set C'.
  2. We compute the inconsistency graph G where the set of vertices is given by $[m]$ and we add an edge (i, j) if Sh_i and Sh_j are pairwise inconsistent.
  3. We set C' to be the minimum vertex cover of this graph.
  4. If $|C'| > \lambda$, we set $\mathsf{msg} = \bot$.

  Via a proof that is identical to Claim 4.8, we can show that Hyb_2 and Hyb_1 are statistically close.

- $\underline{\mathsf{Hyb}_3}$: In this hybrid, we additionally run the rewinding extractor for ECom to obtain $\mathsf{Sh}'_1, \ldots, \mathsf{Sh}'_m$. We check if $|C|$ or $|C'|$ is greater than λ or if any of the checks made by the honest party fails. If that is the case, we set $\mathsf{msg} = \bot$. Otherwise, we set msg as $\mathsf{Rec}(\mathsf{Sh}'_1, \ldots, \mathsf{Sh}'_m)$

  We note that except with negligible probability for every $i \notin C$, Sh_i obtained by SPExt and the one obtained in the Hyb_1 are identical and this follows from the property of the rewinding extractor for ECom. Since $|C \cup C'| < 2\lambda$, it now follows from the pairwise verifiability of the secret sharing scheme that the value reconstructed by in Hyb_2 and Hyb_3 is the same.

- $\underline{\mathsf{Hyb}_4, \mathsf{Hyb}_5}$: In these two hybrids, we reverse the changes made in Hyb_2 and Hyb_1 respectively. Via identical arguments, we can show that $\mathsf{Hyb}_4 \approx_s \mathsf{Hyb}_3$ and $\mathsf{Hyb}_4 \approx_s \mathsf{Hyb}_5$. We note that Hyb_5 is identically distributed to the output of the $(\mathsf{Ext})^{\mathcal{A}}(1^\lambda)$.

□

B Definitions of 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.

B.1 Defining Security.

We assume that readers are familiar with standard simulation-based definitions of secure multi-party 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 B.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 an expected 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 $z \in \{0,1\}^*$.

We specialize this definition to the setting of (multi-party) simultaneous m-choose-ℓ OT. We consider a specific function $f_{OT,n,m,\ell}$, which represents the pairwise m-choose-ℓ ideal OT functionality implemented between n parties. In more detail, for every pair of parties P_i and P_j for $i, j \in [n], i \neq j$, this functionality (implemented by the trusted party) obtains strings $\{x_{\iota,i,j}\}_{\iota \in [m]}$ as well as a choice set $k_{j,i}$ which is a subset of $[m]$ of size ℓ from party P_i. It obtains strings $\{x_{\iota,j,i}\}_{\iota \in [m]}$ and choice set $k_{i,j}$ which is a subset of $[m]$ of size ℓ from party P_j. It outputs $\{x_{\iota,i,j}\}_{\iota \in k_{j,i}}$ to P_j and $\{x_{\iota,j,i}\}_{\iota \in k_{i,j}}$ to P_i.

Definition B.2 (Multi-party OT with Superpolynomial Simulation)**.** *Let k be the security parameter. Let Π be an n-party protocol for $n \in \mathsf{N}$. We say that Π computes simultaneous n-party m-choose-ℓ OT with super-polynomial simulation in the presence of malicious (resp., semi-honest) adversaries if for every PPT adversary (resp., semi-honest adversary) $\mathcal{A}$ there exists a 2^{λ^ϵ} adversary (resp., semi-honest adversary) Sim that makes straight-line, black-box access to $\mathcal{A}$ and interacts with the $f_{OT,n,m,\ell}$ ideal functionality such that for any $I \subset [n]$ with $|I| \leq n-1$ the following quantity is negligible:*

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

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

Remark B.3 (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 security with selective abort.*

Remark B.4 (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.*

B.2 Security Against Semi-Malicious Adversaries

We take this definition almost verbatim from [AJL$^+$12]. 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 PP_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 PP_k up to that point, including the new message m, must exactly match the honest protocol specification for PP_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 PP_k in any step of the interaction.

Definition B.5. *We say that a protocol* Π *securely realizes* f *for semi-malicious adversaries if it satisfies Definition B.1 when we only quantify over all semi-malicious adversaries* A*.*