Round-Optimal Black-Box MPC in the Plain Model

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

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

Abstract. We give the first construction of a (fully) black-box round-optimal secure multiparty computation 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) most 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].[5] 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

[5] 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.

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?

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 [6]

On the role of subexponentially secure OT. We stress that even though we rely on sub-exponentially 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

[6] 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.

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.

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.[7] 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 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.

[7] 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.

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.[8] 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.

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:

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

– 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.[9]
– 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 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

[9] 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].

the corrupted parties. We need to extract this information in order to invoke the security of the outer protocol.[10] 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.

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

[10] 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.

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 receiver 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 $(\mathsf{Enc}, \mathsf{Dec})$.[11] 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.

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 $(\mathsf{m}_b, \mathsf{R}_0, \mathsf{R}_1)$.

Fig. 1: The functionality $\mathcal{F}$

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*

[11] 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.

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.

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 $(\mathsf{m}_{\widetilde{b}}, \mathsf{L}_0, \mathsf{L}_1)$ in the

main thread, and $(\mathsf{m}_{\tilde{\mathsf{b}}}, \mathsf{M}_0, \mathsf{M}_1)$, $(\mathsf{m}_{\tilde{\mathsf{b}}}, \mathsf{R}_0, \mathsf{R}_1)$ 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 $\mathsf{m}_{1-\tilde{\mathsf{b}}}$. Yet, we show that if Π satisfies this property, it becomes possible to replace $\mathsf{m}_{1-\tilde{\mathsf{b}}}$ 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 a construction of 3-out-of-3 non-malleable secret sharing from [GSZ21] 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 $\perp$. 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.

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].

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.

Organization. Due to lack of space, we only present the watchlist protocol and defer the other constructions and their proof of security to the full version.

3 Preliminaries

We recall some standard cryptographic definitions in this section.

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 1 (One-many augmented split-state non-malleable codes). *Fix any polynomials $\ell(\cdot), p(\cdot)$. An $\ell(\cdot)$-augmented non-malleable code with error $\epsilon(\cdot)$ for messages $m \in \{0,1\}^{p(\lambda)}$ consists of algorithms* $\mathsf{NM.Code}, \mathsf{NM.Decode}$ *where*

- $\mathsf{NM.Code}(m) \to (L, M, R)$ *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 $f = (f_1, f_2, \ldots f_{\ell(\lambda)}), g = (g_1, g_2, \ldots g_{\ell(\lambda)}), h = (h_1, h_2, \ldots h_{\ell(\lambda)})$ 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*

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

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

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 [GSZ21] can be proven to satisfy this definition. This is already implicit in [GSZ21] 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 1. *[GSZ21] 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 1 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 2 (Low-Depth Non-Interactive Proofs). *A low-depth non-interactive proof with perfect completeness and soundness for a relation R consists of an (efficient) prover P and a verifier V that satisfy:*

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

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

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

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

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

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 $\mathsf{NC1}$ that must verify validity of a non-malleable code. We rely on low-depth proofs to ensure that the two-party computation protocol only performs $\mathsf{NC1}$ computations.

3.1 3-Round Two-Party Computation Protocol with Special Extraction

The watchlist protocol requires a special two-party computation protocol. We give a construction of this protocol in the full version and present the definition here.

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 $\perp$. 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 $\mathsf{accept/reject}$. 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 3. *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:*

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

In the above definition, we note that if y *output by* Sim_S *is the special symbol* $\perp$*, then the output of* f *is also* $\perp$*. We additionally need the existence of a straight-line SPS simulator* SPSim_S *that has the same guarantees as* Sim_S*.*

– ***Super-Polynomial Time Simulation Security against Malicious Receivers.*** *There exists a super-polynomial time machine* $\mathsf{SPSim}_R = (\mathsf{SPSim}_R^1, \mathsf{SPExt}_R, \mathsf{SPSim}_R^2, \mathsf{SPSim}_R^3)$ *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 2.*

– ***2-Rewinding Sender Security against Sub-Exponential Adversaries.*** *We require that this protocol to be secure against any malicious sub-exponential time receiver that could rewind an honest sender twice by giving possibly different second round messages in each rewind.*

– ***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* $\mathsf{SPExt}_R(\mathsf{msg}_R^1)$ *(where* $\mathsf{msg}_R^1 \in \mathbb{T}$*) 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.*

– ***Delayed Function Selection.*** *The function to be computed can be chosen by the sender in the third round.*

4 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

- Run $\mathsf{SPSim}_R^1(1^\lambda)$ to obtain $(\mathsf{msg}_1^S, \mathsf{st})$ and send msg_1^S to $\mathcal{A}$. Receive msg_1^R from $\mathcal{A}$.
- Run $\mathsf{SPExt}_R(\mathsf{msg}_1^R)$ to obtain x, st'.
- Sample $(\mathsf{msg}_2^S, \mathsf{st}'')$ from $\mathsf{SPSim}_R^2(\mathsf{st}, \mathsf{st}')$ and send msg_2^S to $\mathcal{A}$.
- Receive msg_2^R from $\mathcal{A}$.
- Run $\mathsf{SPSim}_R^3(\mathsf{st}'', f(x,y), \mathsf{msg}_1^R, \mathsf{msg}_2^R, \mathsf{msg}_2^S)$ to obtain msg_3^S and send this to $\mathcal{A}$.
- Receive msg_3^R from $\mathcal{A}$.
- Output view of $\mathcal{A}$.

Fig. 2: Description of Ideal_R.

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.

4.1 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 4 (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 3.
   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.
   (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 3 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* $\{\mathbf{x}_i\}_{i\in M}$*.*

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 $\{\sigma_j\}_{j\in H}$.

Fig. 3: Description of Ideal_{SPS}.

4.2 Construction

Our construction is described in Figure 4, and makes use of the following ingredients:

- A 3 round two-party secure computation protocol Π satisfying Definition 3 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 1.
- A low-depth proof for P according to Definition 2.
- 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 4. 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 4.

Theorem 1. *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 4 that makes black-box use of any 3 round two-party secure computation protocol Π satisfying Definition 3 with 2-rewinding sender security, and any existentially unforgeable signature scheme.*

Proof of Theorem 1. 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 SPS security of the two-party protocol against malicious adversaries (following [IKSS21]). 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 $\mathsf{Real}_{\mathsf{MIM}}\langle\{\mathcal{S}_j(\{\mathsf{m}_{i,j}\}_{i\in[m]})\}_{j\in[\ell]}, \{\mathcal{R}_j(K_j)\}_{j\in[\ell]}$

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 $\perp$.

Fig. 4: $\ell(\lambda)$ Non-Malleable $m(\lambda)$-choose-$k(\lambda)$ Oblivious Transfer

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:

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 $\gamma(\lambda)=\max\left(\lambda,p^{-2}(\lambda)\right)$. 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 $\widetilde{\mathsf{m}}^j_i=\perp$ for all $i\in[m]$. Now, the output of this hybrid is the joint distribution

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

Lemma 2. *For every unbounded distinguisher $\mathcal{D}$ and large enough $\lambda\in\mathbb{N}$,*

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

Proof. Since the MIM's inputs $\{\widetilde{\mathsf{m}}^j_i\}_{j\in[\ell]}$ 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 $\mathsf{negl}(\lambda)$.

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 $\mathsf{Sim\text{-}2PC}^{\mathsf{MIM},\mathcal{F}(\mathsf{inp}_{\mathcal{S}^j},\cdot)}_{\mathsf{Sen}}$ 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 $c_1^j, \ldots, c_\lambda^j$ 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 $(\mathsf{c}_1^j, \ldots, \mathsf{c}_\lambda^j)$ 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 $(\mathsf{c'}_1^j, \ldots, \mathsf{c'}_\lambda^j)$ and $(\mathsf{c''}_1^j, \ldots, \mathsf{c''}_\lambda^j)$ respectively. For every $j \in [\ell]$, let index $\alpha_j \in [\lambda]$ be such that $\mathsf{c}_{\alpha_j}^j = 0, \mathsf{c'}_{\alpha_j}^j = 1, \mathsf{c''}_{\alpha_j}^j = 2$. Additionally for every $j \in [\ell], i \in [m]$, use $(\widetilde{\mathsf{L}}_{\alpha_j,i}^j, \widetilde{\mathsf{M}}_{\alpha_j,i}^j, \widetilde{\mathsf{R}}_{\alpha_j,i}^j)$ obtained as output from the main and the two rewinding executions respectively to compute $\widetilde{\mathsf{m}}_i^j = \mathsf{NM.Decode}(\widetilde{\mathsf{L}}_{\alpha_j,i}^j, \widetilde{\mathsf{M}}_{\alpha_j,i}^j, \widetilde{\mathsf{R}}_{\alpha_j,i}^j)$. If no such rewinding thread exists, or if there exists $j \in [\ell]$ for which there does not exist $\alpha \in [\lambda]$ such that $\mathsf{c}_\alpha^j = 0, \mathsf{c'}_\alpha^j = 1, \mathsf{c''}_\alpha^j = 2$, then abort. The output of this hybrid is the joint distribution:

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

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

Lemma 3. *Assuming 2-rewinding secure two party computation according to Definition 3, for every PPT distinguisher $\mathcal{D}$ and large enough $\lambda \in \mathbb{N}$,*

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

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 $\{m_i^j\}_{i\in[m]}$, 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

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

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

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

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 $\widetilde{\mathsf{m}}^j_i = \perp$ 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 $\lambda \in \mathbb{N}$,

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

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

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

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

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

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 4. *Assuming $m(\lambda)\cdot\ell(\lambda)$ symmetric non-malleable codes satisfying Definition 1, for every unbounded distinguisher $\mathcal{D}$ and large enough $\lambda \in \mathbb{N}$,*

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

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

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

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 $(\overline{\mathsf{c}}_1, \ldots, \overline{\mathsf{c}}_n)$ 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 $\widetilde{\mathsf{c}}_{k^*}, \overline{\mathsf{c}}_{k^*}$ 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 $\mathsf{Sim\text{-}2PC}_{\mathsf{Sen}}$ using randomness r on out to generate the third round message of the protocol transcript in the thread corresponding to the receiver challenge being 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 $\mathsf{Sim\text{-}2PC}_{\mathsf{Sen}}$ 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 $\mathsf{Sim\text{-}2PC}_{\mathsf{Sen}}$ 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 $\mathsf{Sim\text{-}2PC}_{\mathsf{Sen}}$ 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 1 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 $H_{i,j}$,

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

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

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 $\widetilde{c}^{j^*}_{k^*}, \overline{c}^{j^*}_{k^*}$ and $\widehat{c}^{j^*}_{k^*}$ are all different. On the other hand, when any two of the three values $\widetilde{c}^{j^*}_{k^*}, \overline{c}^{j^*}_{k^*}$ and $\widehat{c}^{j^*}_{k^*}$ 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 extends to show that 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.

5 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 senders. Specifically, we prove the following theorem.

Theorem 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 four-round protocol for computing general functions.*

5.1 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 4. 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 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. By (T, ϵ)-security, we require ϵ distinguishing advantage against any adversary that runs in time T. By privacy with knowledge of outputs [IKP10], we consider a weaker notion of security (when compared to standard malicious security in the real/ideal paradigm), wherein the adversary has the additional power to determine the outputs of the honest parties. This is modelled by the ideal functionality getting an output to be delivered to the honest parties from the adversary. 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 5 (discussed below) against adversaries running in time $(T_1 + T_2) \cdot \mathsf{poly}(\lambda)$ and the distinguishing advantage being $\mathsf{negl}(\lambda)$.

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.

Definition 5 ([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.
- ***Security.*** *Let $\mathcal{A}$ be an adversary corrupting a subset of the parties indexed by the set M and let H be the set of indices denoting the honest parties. We require the existence of a simulator Sim_Π such that for any choice of honest parties inputs $\{x_i\}_{i \in H}$, we have:*

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

  where the real and ideal experiments are described as in [IKSS21] (details deferred to the full version due to lack of space).

Given these building blocks, our construction is described in Figure 5.

Due to space constraints, the full proof of security of this construction is deferred to the full version.

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.

- **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.

Fig. 5: Description of the Four-Round MPC Protocol

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.

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.

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.

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.

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.

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.

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.

GSZ21. Vipul Goyal, Akshayaram Srinivasan, and Chenzhi Zhu. Multi-source non-malleable extractors and applications. In *EUROCRYPT 2021, Part II*, pages 468–497, 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, *EUROCRYPT 2020*, 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.

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.

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.