List Oblivious Transfer and Applications to Round-Optimal Black-Box Multiparty Coin Tossing

Citation for published version:
Ciampi, M, Ostrovsky, R, Siniscalchi, L & Waldner, H 2023, List Oblivious Transfer and Applications to Round-Optimal Black-Box Multiparty Coin Tossing. in *CRYPTO 2023: 43rd International Cryptology Conference.* vol. 14081, Lecture Notes in Computer Science, vol. 14081, Springer, pp. 459-488, 43rd International Cryptology Conference, Santa Barbara, California, United States, 19/08/23. https://doi.org/10.1007/978-3-031-38557-5_15

Digital Object Identifier (DOI):
https://doi.org/10.1007/978-3-031-38557-5_15

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
CRYPTO 2023

List Oblivious Transfer and Applications to Round-Optimal Black-Box Multiparty Coin Tossing

Michele Ciampi[1], Rafail Ostrovsky[2], Luisa Siniscalchi[3], and Hendrik Waldner[4,5]

[1] The University of Edinburgh, Edinburgh, UK
michele.ciampi@ed.ac.uk

[2] University of California, Los Angeles, US
rafail@cs.ucla.edu

[3] Technical University of Denmark, Copenhagen, Denmark
luisi@dtu.dk

[4] University of Maryland, College Park, US
hwaldner@umd.edu

[5] Max Planck Institute for Security and Privacy, Bochum, Germany

Abstract. In this work we study the problem of minimizing the round complexity for securely evaluating multiparty functionalities while making black-box use of polynomial time assumptions. In Eurocrypt 2016, Garg et al. showed that, assuming all parties have access to a broadcast channel, then at least four rounds of communication are required to securely realize non-trivial functionalities in the plain model. A sequence of works follow-up the result of Garg et al. matching this lower bound under a variety of assumptions. Unfortunately, none of these works make black-box use of the underlying cryptographic primitives. In Crypto 2021, Ishai, Khurana, Sahai, and Srinivasan came closer to matching the four-round lower bound, obtaining a five-round protocol that makes black-box use of oblivious transfer and PKE with pseudorandom public keys.

In this work, we show how to realize any input-less functionality (e.g., coin-tossing, generation of key-pairs, and so on) in *four rounds* while making black-box use of two-round oblivious transfer. As an additional result, we construct the first four-round MPC protocol for generic functionalities that makes black-box use of the underlying primitives, achieving security against non-aborting adversaries.

Our protocols are based on a new primitive called *list two-party computation*. This primitive offers relaxed security compared to the standard notion of secure two-party computation. Despite this relaxation, we argue that this tool suffices for our applications. List two-party computation is of independent interest, as we argue it can also be used for the generation of setups, like oblivious transfer correlated randomness, in three rounds. Prior to our work, generating such a setup required at least four rounds of interactions or a trusted third party.

Table of Contents

1 Introduction

Secure multiparty computation (MPC) [Yao86, GMW87] allows a group of mutually distrustful parties to jointly evaluate any function over their private inputs in such a way that no one learns anything beyond the output of the function. Since its introduction, MPC has been extensively studied in terms of assumptions, complexity, security definitions, and execution models [GMW87, Kil88, IPS08, BMR90, KOS03, KO04, Pas04, PW10, Wee10, Goy11, GMPP16, ACJ17, BHP17, COWZ22, CRSW22]. In [GMPP16] Garg et al. established that *four rounds* are necessary to securely realize non-trivial functionalities (e.g., coin tossing) in the plain model with static corruption and black-box simulation[6] in the dishonest majority setting. A sequence of works tried to match this lower bound for the multiparty setting [ACJ17, BHP17], and finally [BGJ$^+$18, HHPV18] showed the first four-round protocols based on polynomial time number theoretic assumptions. This result was later improved in [CCG$^+$20] proposing a four-round scheme based on malicious secure oblivious transfer (OT).

Unfortunately, all these round-optimal protocols make non-black-box use of the underlying primitives. Recently, Ishai, Khurana, Sahai, and Srinivasan [IKSS21] came close to closing this gap, proposing a five-round protocol that makes black-box use of oblivious transfer and public-key encryption (PKE) with pseudorandom public keys. Prior to their work, MPC protocols that use the underlying primitives in a black-box way needed more than 15 rounds [IPS08, Wee10, Goy11]. This leaves open the following questions:

What is the best-possible round complexity for securely evaluating any multiparty functionality in the plain model when the majority of the parties are corrupted while using the underlying cryptographic primitives in a black-box way?

1.1 Our Contribution

In this work, we partially answer the above question by constructing the first round-optimal MPC protocol for computing any input-less functionality (e.g., coin tossing), while making black-box use of any perfect correct two-round private OT protocol[7]. The notion of private OT, introduced in [NP01, AIR01], requires that the receiver's only message is computationally indistinguishable between the cases where its input bit b is equal to 0 or 1. Moreover, regardless of the receiver's first message, the sender's message hides at least one of its inputs (denoted by s_0 and s_1). Private OT can be obtained from LWE [BD18], or number theoretic assumptions such as DDH [NP01, AIR01] and DCR [Kal05]. As an additional contribution, we show how to securely realize *any* efficiently computable functionality in four rounds, while relying on the same assumptions, against a weaker form of adversaries that promises to not abort during the computation with some unknown non-negligible probability. We will refer to this type of adversary as *sometimes aborting.* In summary, we prove the following theorems.

Theorem 1 (informal). *Assuming two-round private OTs, then there exists a* 4*-round MPC protocol that realizes any multiparty input-less functionality with black-box use of the underlying primitives.*

Theorem 2 (informal). *Assuming two-round private OTs, then there exists a* 4*-round MPC protocol that realizes any multiparty functionality with black-box use of the underlying primitives in the presence of sometimes aborting adversaries.*

Our protocols are based on a new primitive called *list two-party computation.* This notion is an extension of the definition of list coin-tossing introduced in [BGJ$^+$18]. List coin tossing relaxes the standard simulation-based security definition, by allowing the adversary (hence, the simulator) to query the ideal functionality multiple times, thus obtaining multiple outputs (uniform random strings in this case). The adversary can then force the output of the honest parties to be one of the strings received from the ideal functionality.

[6] All the results discussed and presented in this paper are with respect to black-box simulation. Hence, we will not specify this in the remainder of the paper.

[7] In a concurrent and independent work [IKSS23], the authors show a 4-round protocol for any functionality relying on assumptions secure against sub-exponential time adversaries. In our work, we make only use of polynomial time assumptions while focusing on the class of input-less functionalities.

List two-party computation (2PC) extends the notion of list coin-tossing to generic functionalities. In particular, we start by considering the *list OT functionality*. In this setting, for a corrupted receiver, the ideal world is formalized as follows: the adversary provides its input b and a parameter κ to the ideal functionality. The ideal functionality then samples κ random pairs $(s_0^i, s_1^i)_{i\in\kappa}$ uniformly at random, and provides $(s_b^i)_{i\in k}$ to the adversary. The adversary now picks an index $j \in [\kappa]$ and sends it to the ideal functionality, who delivers (s_0^j, s_1^j) to the sender in the ideal world. In summary, we allow the adversarial receiver to slightly bias the output of the computation. In [BGJ+18] the authors observe that list coin-tossing suffices to generate a common random setup (e.g., for non-interactive zero-knowledge arguments). Similarly, we can argue that list-OT is sufficient for the generation of correlated OT randomness.

As previously mentioned, we propose a notion that is generic for any two-party functionality f. Suppose that only one party receives the output, so we can distinguish between a sender S and a receiver R, and let us assume that R is corrupted. The ideal functionality, upon receiving the input of the corrupted party x_R and κ, samples for each $i \in [\kappa]$ an input (according to some distribution by which the ideal world is parametrized) from the input domain of the sender, obtaining x_S^i. Afterwards, it computes $y^i \leftarrow f(x_R^i, x_S^i)$ and delivers $\{y^i\}_{i\in[k]}$ to the adversary. Upon receiving j from the adversary, the functionality returns (x_R^j) to the ideal world sender.

We show how to instantiate a list two-party protocol in just three rounds, where, in each round, both of the parties speak at the same time (this communication model is often referred to as the simultaneous message model), while making black-box use of the underlying cryptographic primitives. This tool will form the basis to instantiate our 3-round list-simulatable non-malleable OT protocol, which we will then plug into the MPC protocol proposed in [IKSS21] to obtain the first 4-round MPC protocol for input-less functionalities. We give more details on this in the next section. Additionally, we also show that a slight modification of this protocol yields the first round-optimal MPC protocol that realizes any functionality against sometimes aborting adversaries. We finally note that our list-OT protocol can be used to generate OT-correlated randomness in just three rounds. Prior results, required the execution of a four-round protocol or needed to rely on a trusted party for generating this type of setup. Closing the gap completely, and showing how to obtain a round optimal MPC protocol for generic functionalities from polynomial time assumptions while making black-box use of the underlying primitives, remains an important open question.

1.2 Technical overview

List oblivious transfer. The first tool we construct, which will form the base of our three-round 2PC protocol, is a list OT protocol. Our protocol is based on a two-round private OT protocol which we denote with $\mathsf{OT}^{\mathsf{priv}}$. Our list OT protocol, which we denote with $\mathsf{OT}^{\mathsf{list}}$, works as follows: the sender and receiver run an OT combiner[8] on two instances of $\mathsf{OT}^{\mathsf{priv}}$. In the second round, the sender tosses a random coin c and asks the receiver to send a valid defence[9] for the c-th execution of $\mathsf{OT}^{\mathsf{priv}}$. The receiver sends the defence to the sender, which accepts if the defence is valid. The security of the combiner guarantees that the input choice of the receiver remains protected, even if the randomness of one of the OT instances is leaked. Hence, the obtained protocol preserves the privacy of the receiver's input. To prove that the protocol is list simulatable against corrupted receivers, we construct a simulator that rewinds the second round by sending a new coin $c' \neq c$. Upon receiving the defence for the remaining OT protocol, the simulator is able to reconstruct the input of the receiver.

There are many aspects that have not been taken into account in this high level description. For example, the list simulator needs to receive a valid third round to complete the simulation. If the corrupted receiver is rushing, then the simulator needs to complete an entire execution of the protocol before starting the extraction procedure. To do this, the simulator uses random inputs, during the main thread and the rewinding

[8] We recall that an OT combiner allows combining two (or more) OT instances to obtain a new OT instance, that is guaranteed to be secure, as long as one of the input instances is not compromise. We refer to [HKN+05] for more details on OT combiners. Moreover, for our formal constructions we will not rely on a combiner in a black-box way, we will instead use techniques similar to those proposed in [HKN+05] to properly combine our OT protocols.

[9] A defence of a protocol execution is represented by the randomness and input that explain the generated transcript.

threads, and behaves as an honest sender would. After the extraction is performed, the simulator can query the ideal functionality with the choice bit of the receiver b and obtains as a reply s_b, which can then be used in the simulation. In more detail, the simulator rewinds the adversary and uses the input (s_b, s_b) to execute the OT combiner as the honest sender would. This approach is still unsatisfactory since it encounters the same issue as observed in [IKSS21]. Indeed, it may be that the receiver aborts when receiving the output s_b, whereas it did not abort in the main thread (where the simulator may have used $s'_b \neq s_b$). This is the reason why Badrinarayanan et al. [BGJ$^+$18] considered a relaxed ideal functionality, that returns many possible outputs to the adversary (and the simulator), which gives the simulator multiple opportunities to force the output to the receiver.

There are two main aspects left to clarify. The protocol just described works under the assumption that the input of the sender is sampled uniformly at random from a sampling space of exponential size (or at least with high min-entropy). This condition is necessary to prove the security of all the list-protocols we propose in this work. We also note that the protocol just described is still not secure. A receiver could decide to complete the protocol when $c = 0$ and abort when $c = 1$. Against such an adversary the simulator clearly fails. To solve this problem, the sender performs an n-out-of-n (where n is polynomial in the security parameter) secret sharing of its two inputs, thus obtaining $(s_0^1, \ldots, s_0^n)$, $(s_1^1, \ldots, s_1^n)$. Then, the sender and the receiver engage in n repetitions of the described protocol, where the sender uses as its input to the i-th combiner the values (s_0^i, s_1^i), and the receiver uses its bit b. We can now argue that the simulator will successfully extract from at least one of the executions after an (expected) polynomially many rewinds. We highlight that if the corrupted receiver uses different inputs in different executions of the combiner, the receiver will not obtain enough shares to reconstruct neither s_0 nor s_1.

Unfortunately, we are only able to prove the property of list simulation against an adversary that does not abort in the third round. If a corrupted receiver does abort, then there is no hope of extracting its input. We can still argue that, in this case, the privacy of the sender's input remains, in the same way as for $\mathsf{OT}^{\mathsf{priv}}$.

Another crucial property we require from our OT protocol is *B-rewind security* (we elaborate more on the reason why we require it later in this section). This property guarantees that all the parties' inputs remain protected even if the adversary is able to rewind the receiver B times, thus getting multiple valid third rounds computed with respect to $B + 1$ different second rounds. Our protocol $\mathsf{OT}^{\mathsf{list}}$ trivially does not achieve this property in the case of a corrupted sender (i.e., a corrupted sender could act as the list-simulator and extract the receiver's input). To achieve 1-rewind security, we iterate the idea of the combiners, twice. More precisely, in each of the n iterations, we consider four executions of $\mathsf{OT}^{\mathsf{priv}}$ (let us denote each instance as *leaf*), and apply a combiner on the first two leaves, therefore obtaining a new OT protocol (which we call *left node*). We do the same for the second two leaves thus obtaining a new OT protocol (denoted by *right node*). Afterwards, we apply a combiner on the two nodes (obtaining the root). The sender now sends a challenge $c \in \{0, 1\}^2$ that determines how to navigate the constructed tree. In more detail, the first bit of c selects the node, and the last bit selects the leaf. For the selected leaf, the receiver will provide the defence. Now, even if a corrupted sender rewinds the receiver, the best it can achieve is to extract 2 out of 4 defenses among the four different executions of $\mathsf{OT}^{\mathsf{priv}}$, and the security achieved by the combiners protects the input of the receiver. To increase the rewind security of the protocol, we can iterate the above approach recursively.

To construct a list protocol for generic functionalities, we use the 1-out-of-2 OT protocol just described, and a (B-rewind secure) k-out-of-n list OT protocol. To obtain such a protocol, we devise a compiler inspired by [SSR08] that turns our 1-out-of-2 OT protocol into a k-out-of-n OT protocol while preserving the round complexity and all the security properties of the underlying 1-out-of-2 protocol. In the remainder of this section, we denote by $\mathsf{OT}_{1,2}^{\mathsf{ListRew}}$ and $\mathsf{OT}_{k,n}^{\mathsf{ListRew}}$ the 1-out-of-2 and the k-out-of-n list-oblivious transfer B-rewind secure protocols, respectively.

List two-party computation. The main structure of our two-party protocol is the same as the one proposed in [IKSS21]. The protocol of [IKSS21] can be seen as being composed of two main components: a four-round oblivious transfer protocol, and an application of the IPS [IPS08] compiler that uses the OT to perform a cut-and-choose (this is often referred as the watchlist mechanism). The main reason why the 2PC protocol of [IKSS21] requires four-rounds is due to the use of a simulation based secure oblivious transfer protocol. Indeed, to obtain such an OT protocol four rounds of communication are necessary.

At a very high level, to obtain our three-round protocol we replace the 4-round OT protocol with a 3-round OT protocol, that is simulation based secure against corrupted senders, and list simulatable against corrupted receivers. Our main contribution is to construct such a 3-round OT protocol. We now give a sketch of this protocol.[10]

To achieve our goal, we need to enhance the security of $\mathsf{OT}^{\mathsf{ListRew}}_{1,2}$, and make it simulation based secure against corrupted senders, while preserving the list simulatability against corrupted receivers. To do that, we need a way to force the sender to compute its messages honestly and employ a mechanism that allows the extraction of the sender's inputs. The general idea to achieve this is the following: we let the sender and the receiver engage in multiple executions of $\mathsf{OT}^{\mathsf{ListRew}}_{1,2}$, and run a cut-and-choose to check that the sender is behaving correctly in a big portion of the OT executions (for this to work, the input of the sender needs to be properly secret shared among the executions. For simplicity, we will ignore this aspect in this overview). Unfortunately, this approach does not work in our case, because we only have a total of three rounds, and the first message of the sender of $\mathsf{OT}^{\mathsf{ListRew}}_{1,2}$ appears in the second round. Hence, there is no room for the cut-and-choose (if the receiver sends the challenge in the first or the second round the sender can adaptively decide which execution to perform correctly and which to perform incorrectly). To solve this problem we hide the challenge of the cut-and-choose using our k-out-of-n OT protocol $\mathsf{OT}^{\mathsf{ListRew}}_{k,n}$. The protocol to hide the challenge is run in parallel with the executions of $\mathsf{OT}^{\mathsf{ListRew}}_{1,2}$. In this protocol, the receiver inputs a random set of indices denoted with K, where $|K| = k$ (with k being a parameter that depends on the cut-and-choose), and the sender uses as its input the n defences for the executions of $\mathsf{OT}^{\mathsf{ListRew}}_{1,2}$. At the end of the protocol, the receiver accepts the output only if all the k defences obtained from $\mathsf{OT}^{\mathsf{ListRew}}_{k,n}$ are valid. Intuitively, this approach should guarantee that a big portion of the 1-out-of-2 OT executions are correct. Unfortunately, it is not clear how this can be formally argued. Indeed, one would like to claim that if this does not hold (i.e., the majority of the 1-out-of-2 OTs are not correct), then we can construct a reduction that breaks the receiver privacy of $\mathsf{OT}^{\mathsf{ListRew}}_{k,n}$.

The reason why this is non-trivial is that the reduction would need to check in which of the $\mathsf{OT}^{\mathsf{ListRew}}_{1,2}$ executions the sender is behaving correctly, and this would reveal information about what input is used by the challenger (which is acting as the receiver) of $\mathsf{OT}^{\mathsf{ListRew}}_{k,n}$ in the reduction. For example, if we could detect that the sender's messages in the i-th execution of $\mathsf{OT}^{\mathsf{ListRew}}_{1,2}$ are not well-formed, then we know that $i \notin K$ with some non-negligible probability. The problem is that the reduction cannot efficiently check which executions are correct and which executions are not. To enable an efficient check, we modify the protocol requiring the sender to commit to the defence via a three-round extractable commitment scheme. We also require the sender, to input to the $\mathsf{OT}^{\mathsf{ListRew}}_{k,n}$ the randomness used to compute the extractable commitments. Hence, the honest receiver accepts the output if the randomness obtained via $\mathsf{OT}^{\mathsf{ListRew}}_{k,n}$ is valid with respect to the extractable commitments, and, moreover, the messages committed are valid defences for k executions of $\mathsf{OT}^{\mathsf{ListRew}}_{1,2}$. The reduction to the receiver security of $\mathsf{OT}^{\mathsf{ListRew}}_{k,n}$ can now extract from the extractable commitments, and check which $\mathsf{OT}^{\mathsf{ListRew}}_{1,2}$ executions are executed honestly by the sender. This information can then be used by the reduction to infer what is the value K is used by the challenger, thus reaching a contradiction. We note that this reduction crucially relies on the rewind security of $\mathsf{OT}^{\mathsf{ListRew}}_{k,n}$. Indeed, we need to make sure that we can extract from the extractable commitments while not perturbing the reduction to the receiver security of $\mathsf{OT}^{\mathsf{ListRew}}_{k,n}$. The addition of the extractable commitments helps us also in designing the simulator, allowing the simulator to extract the sender's input using the extractable commitments.

Now, we have constructed a three-round protocol that is simulation based secure against corrupted senders. We further need to make sure that this protocol is list-simulatable against corrupted receivers. At a high level, the list simulator runs the list simulators for $\mathsf{OT}^{\mathsf{ListRew}}_{k,n}$ and $\mathsf{OT}^{\mathsf{ListRew}}_{1,2}$, where the input extracted by the simulator of $\mathsf{OT}^{\mathsf{ListRew}}_{1,2}$ defines the choice bit of the receiver, and the values returned by the simulator of

[10] Our formal construction directly uses $\mathsf{OT}^{\mathsf{list}}$ to obtain a 3-round list 2PC protocol, so we do not need an intermediate step where we instantiate this special OT protocol, which is instead implicit in our 2PC protocol.

$\mathsf{OT}_{k,n}^{\mathsf{ListRew}}$ help to program which of the extractable commitments must contain a valid defence, and which commitments instead should contain a dummy value (e.g., the all-zero string). There are two subtleties in this simulation strategy. First, once K is extracted, the extractable commitments need to be programmed accordingly. Unfortunately, this is not possible if the extractable commitments are binding in the first round. This problem can be easily solved by using a delayed-input extractable commitment scheme. The other problem is related to the fact that in the security proof we need to rely on the hiding of the extractable commitments. In particular, we would need to rely on the hiding property while we are running the simulator of $\mathsf{OT}_{1,2}^{\mathsf{ListRew}}$, hence, we need an extractable commitment that is secure against rewinds. This creates a circularity, since we already require $\mathsf{OT}_{1,2}^{\mathsf{ListRew}}$ to be 1-rewind secure, but 1-rewind security is already sufficient to extract from the extractable commitment. Furthermore, we want to extract the input by rewinding $\mathsf{OT}_{1,2}^{\mathsf{ListRew}}$ while maintaining the hiding of the extractable commitment. To break this circularity, we rely on an extractable commitment that has a simulatable third round [HHPV21]. In a nutshell, this property allows to simulate a third round of the extractable commitment without knowing the randomness used to compute the first message. This property breaks the circularity, but it inherently makes the commitment extractable with over-extraction, hence, some effort is needed to show that all the arguments we have summarized in this section still hold.

Unfortunately, our 2PC has a limitation. The proof against corrupted receivers crucially relies on the fact that the value K can be extracted. This indicates which defences of the executions of $\mathsf{OT}_{1,2}^{\mathsf{ListRew}}$ the adversary will not obtain, and allows us to rely on the sender security of $\mathsf{OT}_{1,2}^{\mathsf{ListRew}}$. If a corrupted sender aborts in the third round, the extraction of K is not possible, which results in the simulator being stuck. For this reason, we prove that our 2PC enjoys a relaxed form of security, which guarantees that the simulation succeeds only when the corrupted sender provides an accepting transcript in the main-thread. Despite this limitation, we show that our 2PC suffices for our main applications.

List non-malleable OT. The notion of non-malleable oblivious transfer has been introduced in [IKSS21]. In this notion, there is a man-in-the-middle adversary MiM that acts as a receiver in a subset of OT sessions (referred to as the *left sessions*) and as a sender in a different subset of OT sessions (referred to as the *right sessions*). A non-malleable OT requires the input used by the MiM acting as the sender being independent of the input used by the honest senders in the left sessions. This is formalized by requiring the existence of a simulator-extractor, that, given the inputs of all honest receivers participating in the right sessions, is able to extract all the implicit inputs used by the MiM in its right sessions. Note that the simulator does not have access to the inputs of the senders in the left session.

We recall how the Ishai et al. (four-round) construction, which we denote by $\mathsf{OT}^{\mathsf{NM}}$, works and then explain how we squeeze it into three rounds allowing us to obtain a list simulatable version of the protocol. The 1-out-of-2 NM-OT protocol of [IKSS21], uses a 4-round simulation secure two-party computation protocol Π that enjoys some special properties (we will elaborate more on this later) and a two split-state non-malleable code $\mathsf{NM} = (\mathsf{Code}, \mathsf{Decode})$.[11] The sender of $\mathsf{OT}^{\mathsf{NM}}$ uses inputs L_0, R_0 and L_1, R_1 which are obtained by running Code on s_0 and s_1 respectively (where s_0 and s_1 represent the inputs of the OT sender). The receiver of $\mathsf{OT}^{\mathsf{NM}}$ uses as its input the choice bit b. Then sender and receiver execute Π for the functionality f_c, where c is a random bit chosen by the receiver in the third round. f_c takes as input $L_0, R_0, L_1, R_1, s_0, s_1, c, b$ and returns m_b, L_0, L_1 if $c = 0$ and m_b, R_0, R_1 otherwise.

The authors of [IKSS21] require that Π has the following properties: (1) the receiver input b is committed in the first round and security holds even if the receiver can adaptively select the function to be computed (i.e., the parameter c) in the third round; (2) the inputs of the sender are committed in the 2nd round; (3) Π is 1-rewinding sender secure. To argue that non-malleability holds, Ishai et.al. rely on an extractor, that can

[11] A split state non-malleable code has an encoding algorithm Code that, on input a message m returns two codewords L and R. The security of the non-malleable code guarantees that there are no tampering functions f and g, that taking as an input L and R, respectively, return $\tilde{L}$ and $\tilde{R}$, such that the decoding of these tampered codewords has a relation with the message m.

compute all the codewords of the MiM acting as senders in the right sessions, by sending different values of c through multiple rewinds.

In our approach, we follow a similar blueprint but use our three-round two-party computation protocol Π that satisfies the notion of list simulatability. We can prove that our 2PC also satisfies properties (1) and (3) mentioned above. However, the input of the sender in our Π is fixed in the last round. Hence the inputs of the sender can change during the rewinds executed to extract from MiM. Therefore, we modify the non-malleable OT protocols as follows: We require the sender to commit via a three rounds non-malleable commitment scheme to each codeword. We then modify the function computed by the 2PC protocol in such a way that when $c = 0$, the opening of the commitment for the left state codewords is sent as an additional output of the 2PC, and, if $c = 1$, the opening of the commitment for the right state codeword is sent. Unfortunately, forcing the sender to commit to their inputs causes problems in the list simulation proof. In particular, we have the same issue we had when designing the 2PC. That is, the list-simulator needs to use random inputs during the extraction phase but if we commit to the codewords, we are implicitly committing to the sender's input in the first round. This prevents the simulator from changing those values after the extraction. To solve this issue, we do not commit to the codeword, instead, we commit to a share of each codeword and output the other share in the clear in the third round. Now, even if the input of the sender is specified in the third round, we have the advantage that the extractor can collect the openings of all the commitments via rewinds. The opened value can then be combined with the shares sent in the clear in the third round of the main thread to finally reconstruct the codewords used by the MiM (in the main thread).

There still remains a single issue. We recall that the receiver specifies the c value in the second round, hence, the MiM could adaptively commit to an invalid codeword by sending a random share instead of the correct codeword share in the third round. We prevent this by hiding the c values as follows: the receiver will send $c \oplus k$, where k is an input given to the 2PC protocol. Hence, the 2PC protocol can compute c and return either the opening of the commitment for the left codeword or the opening of the commitment for the right codeword, but the sender does not know what c has been used by the receiver. We recall that our 2PC protocol Π does not provide any protection of the sender's input if the receiver is aborting. This limitation is inherited by our non-malleable OT protocol. For more details on how our non-malleable OT protocol works, we refer to the technical section of this work.

Four-round multiparty computation. Also in this last step we follow the approach proposed in [IKSS21], which in turns is based on the IPS compiler [IPS08]. This compiler combines an information-theoretic MPC protocol Φ (called *outer protocol*), secure in the honest majority setting, and a semi-honest protocol secure against a dishonest majority Π_h called *inner protocol*.

The inner protocol is used to emulate an execution of Φ, which guarantees that the messages of Φ are generated honestly, as long as the parties running Π_h behave semi-honestly. To guarantee security in the malicious setting, the IPS compiler requires to perform a cut-and-choose to check that the majority of the parties are running Π_h honestly. This cut-and-choose is performed by relying on correlated OT randomness. In [IKSS21] the authors manage to make this paradigm work in just 5 rounds, using a 4-round inner protocol, and a 4-round non-malleable OT protocol for the generation of the correlated randomness.

In this work, we follow the same approach by relying on our three-round non-malleable OT protocol, and on a three-round semi-honest MPC protocol to realize the inner protocol. The main difference is that we are able to prove the security of the MPC protocol only against non-aborting adversaries. But, as mentioned before, this still suffices to securely realize any input-less functionality with standard simulation based security. Furthermore, our inner-protocol also needs to achieve our a special security notion in which the inputs of the honest parties are sampled from a certain distribution by the ideal functionality in a similar fashion as in our list-2PC definition. This modification, with respect to the inner-protocol of [IKSS21], is needed to allow for later equivocality of the simulator to ensure that the overall protocol can be simulated in four-rounds.

2 Preliminaries

Notation. We let $\text{View}^{B}_{\Pi,A}(\lambda, x, y)$ be the random variable corresponding to the *view* of A in an execution of Π with B where A uses x as its input and B uses y as its input. If an algorithm A outputs a transcript τ, then we call $\mathsf{Defense}$ the input that explains the output, i.e. $\tau = A(\mathsf{Defense})$. We denote the security parameter with $\lambda \in \mathbb{N}$. A randomized algorithm $\mathcal{A}$ is running in *probabilistic polynomial time* (PPT) if there exists a polynomial $p(\cdot)$ such that for every input x the running time of $\mathcal{A}(x)$ is bounded by $p(|x|)$. We use "=" to check equality of two different elements (i.e. $a = b$ then...) and ":=" as the assigning operator (e.g. to assign to a the value of b we write $a := b$). A randomized assignment is denoted with $a \leftarrow A$, where A is a randomized algorithm and the randomness used by A is not explicit. If the randomness is explicit we write $a := A(x; r)$ where x is the input and r is the randomness. When it is clear from the context, to not overburden the notation, we do not specify the randomness used in the algorithms unless needed for other purposes.

2.1 t-out-of-n Secret Sharing

Now, we present the definition of a t-out-of-n secret sharing as it has been presented in [MOSV22].

A threshold secret sharing scheme allows to split a secret $s \in \{0,1\}^\lambda$ into n shares $s_1, \ldots, s_n \in \{0,1\}^\lambda$ in such a way that it is possible to efficiently recover s from any subset of at least t shares, while, at the same time, not revealing anything about the secret as long as an attacker only corrupts up to $(t-1)$ of the share holders. More formally:

Definition 2.1 (t-out-of-n Secret Sharing). *A (n,t)-secret sharing scheme over $\{0,1\}^\lambda$ is defined by a pair of algorithms* $(\mathsf{Share}, \mathsf{Rec})$*, where* Share *is a randomized algorithm taking as input $s \in \{0,1\}^\lambda$ and outputs the shares $(s_1, \ldots, s_n) \in (\{0,1\}^\lambda)^n$, and* Rec *is a function mapping a subset $\mathcal{I}$ of $[n]$, along with the corresponding shares $s_\mathcal{I} = (s_i)_{i\in\mathcal{I}}$, to a value in $\{0,1\}^\lambda$, such that the following holds:*

Reconstruction: *For all $s \in \{0,1\}^\lambda$, and for all sets $\mathcal{I} \subseteq [n]$ with $|\mathcal{I}| \geq t$, the output of* $\mathsf{Rec}(\mathcal{I}, s_\mathcal{I})$ *such that $(s_1, \ldots, s_n) \leftarrow \mathsf{Share}(s)$ is equal to s.*

Security: *For all $s \in \{0,1\}^\lambda$, and for all sets $\mathcal{I} \subseteq [n]$ with $|\mathcal{I}| < t$, the joint distribution $s_\mathcal{I} = (s_i)_{i\in\mathcal{I}}$ of shares received by the subset of parties $\mathcal{I}$, where $(s_1, \ldots, s_n) \leftarrow \mathsf{Share}(s)$, is independent of the secret s.*

2.2 Extractable Commitments

In this section, we present the definition of an extractable commitment scheme as well as the notion of delayed inputness.

Definition 2.2 ([BGJ+18]). *Consider any statistically binding, computationally hiding commitment scheme (C, R). Let $\mathsf{Tr}\langle C(m, r_C), R(r_R)\rangle$ denote a commitment transcript with committer input m, committer randomness r_C and receiver randomness r_R, and let $\mathsf{Decom}(\tau, m, r_C)\rangle$ denote the algorithm that on input a commitment transcript τ, committer message m and randomness r_C outputs 1 or 0 to denote whether or not the decommitment was accepted (we explicitly require the decommitment phase to not require receiver randomness r_R). Then $\langle C, R\rangle$ is said to be extractable if there exists an expected PPT oracle algorithm* Ext*, such that for any PPT cheating committer C^* the following holds. Let $\mathsf{Tr}\langle C^*, R(r_R)\rangle$ denote the transcript of the interaction between C^* and R. The extractor* Ext *on input $\mathsf{Tr}\langle C^*, R(r_R)\rangle$ and with oracle access to C^* outputs m s.t. over the randomness of* Ext *and of sampling $\mathsf{Tr}\langle C^*, R(r_R)\rangle$:*

$$\Pr\left[\,\exists(\tilde{m} \neq m, \tilde{r}_C) \text{ s.t. } \mathsf{Decom}(\tau, \tilde{m}, \tilde{r}_C) = 1\,\right] = \mathrm{negl}(\lambda)$$

We note that this definition does not say anything about the correctness of the extracted value for the case where the commitment is ill-formed (i.e., it does not admit an opening)

Definition 2.3. *A three-round extractable commitment satisfying Definition 2.2 is said to be k-extractable if there exists a PPT extractor algorithm Ext such that, given a set of k well-formed execution transcripts of the commitment scheme where each transcript consists of the same first round sender message, and different second rounds from the receiver, the extractor successfully extracts the value committed in each transcript, except with negligible probability.*

Definition 2.4. *We say that an extractable commitment is delayed input if the committer uses the input message m only in the last round of the protocol.*

Furthermore, we also define the notion of 1-rewind security for a commitment.

1-rewind security. Let (C, R) be a three-round commitment scheme. (C, R) is 1-rewind secure if the following games are indistinguishable for $b = 0$ and $b = 1$, for any pair of messages m_0 and m_1.

Game(b): Compute the first round of the protocol C for a message m_b using randomness r thus obtaining com_1, and send it to the adversary. The adversary sends two challenge messages $(\mathsf{com}_2^0, \mathsf{com}_2^1)$, and the challenger computes a third round, as the honest sender would do with respect to the message m_b (using always randomness r) for the challenges com_2^0 and com_2^1 thus obtaining com_3^0 and com_3^1 respectively. The challenger sends $(\mathsf{com}_3^0, \mathsf{com}_3^1)$ to the adversary.

Rewind Secure Extractable Commitment Scheme. After presenting the different definitions for an extractable commitment scheme, we propose, in Figure 2.1, a delayed-input, 1-rewind secure 3-extractable commitment scheme ExtCom (we refer to Section 2.2 for the formal definitions). This commitment scheme is inspired by [GRRV14, KOS18]. We prove the 3-extractability property as follows.

Figure 2.1: Rewind secure Extractable Commitment.

Public parameters: Prime $q > \lambda$.
Senders' private input: message $m \in \mathbb{F}_q$ (available to the sender only in the third round).

1) **Sender.** Sample $r, s, m_0 \leftarrow \mathbb{F}_q$, compute and send $\mathsf{com} \leftarrow \mathsf{Com}(m_0||r||s)$.
2) **Receiver.** Sample and send $\alpha \leftarrow \mathbb{F}_q$.
3) **Sender** Compute $m_1 \leftarrow m + m_0$ and $a \leftarrow m + \alpha r + \alpha^2 s$, and send (a, m_1).

Let $(\mathsf{com}, \alpha^1, (a^1, m_1^1))$, $(\mathsf{com}, \alpha^2, (a^2, m_1^2))$, $(\mathsf{com}, \alpha^3, (a^3, m_1^3))$ be three accepting transcripts generated by an honest sender of ExtCom. The 3-extractor interpolates $(\alpha^1, a^1), (\alpha^2, a^2), (\alpha^3, a^3)$ thus obtaining a quadratic polynomial. Let $\tilde{m}_0$ be the constant term of this quadratic, then we can compute the committed message in each of the input transcripts as $m^i = \tilde{m}_0 + m_1^i$ for each $i \in [3]$. The 1-rewind security comes from hiding on the non-interactive commitment, and the observation that two accepting transcripts with the same first round and different second round give 0 information to the adversary (other than what the adversary can learn by attacking the hiding of the non-interactive commitment). ExtCom also enjoys the property of *simulatability* [HHPV18]. The simulatable property requires that the following games are indistinguishable for $b = 0$ and $b = 1$. For any pair of messages m_0 and m_1.

Game(b): Sample $r, s, \leftarrow \mathbb{F}$, compute the first round of the protocol ExtCom using $m_b||r||s$ thus obtaining com_1, and send it to the adversary. The adversary sends the second round com_1 and the challenger responds according to m_0, r, s. The simulatability property of ExtCom comes from the hiding property of the non-interactive commitment scheme used to generate the first message. We note that this is similar to the property described (and required) in [HHPV18, GKP17].

2.3 Non-Malleable Commitment Scheme

The following definition is taken verbatim from [GPR16]. Non-malleable commitment is defined using the real/ideal paradigm. In the real interaction, there is a man-in-the-middle adversary MIM interacting with a sender, S, in the left session a receiver R in the right. We denote the various quantities associated with the right interaction as "tilde'd" versions of their left counterparts. So for example, S commits to v in the left interaction while MIM commits to $\tilde{v}$ in the right. Let $\mathsf{View}^v_{\mathsf{MIM}}$ denote a random variable that describes (VIEW, $\tilde{v}$), consisting of MIM's view in the experiment and the value MIM commits to in the right interaction, given that S has committed to v in the left interaction. The ideal interaction is the same, except that S commits to an arbitrary fixed value, say 0, on the left. Let $\mathsf{View}^0_{\mathsf{MIM}}$ be the random variable describing (VIEW, $\tilde{v}$) in this interaction. We will use id-based commitment schemes and we force MIM to use an identity $\tilde{\mathsf{id}}$ on the right which is distinct from id used on the left. We enforce this by stipulating that $\mathsf{View}^0_{\mathsf{MIM}}$ and $\mathsf{View}^v_{\mathsf{MIM}}$ output the special symbol $\perp_{\mathsf{id}}$ when MIM has used the same identity on the right which he has received on the left. This is analogous to the uninteresting case when MIM is simply acting as a channel, forwarding messages from S to R and back. We let $\mathsf{View}^0_{\mathsf{MIM}}(y)$ and $\mathsf{View}^v_{\mathsf{MIM}}(y)$ be the distributions where M gets a string y as an auxiliary input.

Definition 2.5. *(Non-Malleable Commitments). A commitment scheme Π_{NMC} is non-malleable if for every PPT adversary* MIM *and for all v we have* $\{\mathsf{View}^v_{\mathsf{MIM}}(y)\}_{y\in 0,1^\lambda} \approx \{\mathsf{View}^0_{\mathsf{MIM}}(y)\}_{y\in 0,1^\lambda}$

A non-malleable commitment is said to enjoy the *public-coin* property if R at each round simply toss a predetermined number of coins (random challenge) and sends them to S. A non-malleable commitment is said to be *synchronous* if the MIM sends the i-th round message on the right immediately after getting the i-th round message in the left interaction. We note that the synchronous non-malleable commitment scheme construct in [GPR16] satisfies the public-coin property and the simulatability property and makes black-box use of OWPs.

2.4 Private Two-Message Oblivious Transfer

A two-message oblivious transfer protocol consists of a tuple PPT algorithm $\mathsf{OT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ with the following syntax.

– $\mathsf{OT}_1(1^\lambda, \beta)$ takes the security parameter λ and a selection bit β and outputs a message ot_1 and secret state st.
– $\mathsf{OT}_2(1^\lambda, (\nu_0, \nu_1), \mathsf{ot}_1)$ takes the security parameter λ and two inputs $(\nu_0, \nu_1) \in \{0,1\}^{\mathsf{len}}$ (where len is a parameter of the scheme) and a message ot_1. It outputs a message ot_2.
– $\mathsf{OT}_3(1^\lambda, \mathsf{st}, \mathsf{ot}_2, \beta)$ takes the security parameter, the bit β, secret state st and message ot_2 and outputs $\nu_\beta \in \{0,1\}^{\mathsf{len}}$.

Correctness and security are defined as follows.

Definition 2.6 ([BD18]). *A tuple PPT algorithm* $\mathsf{OT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ *is a private-OT scheme id the following hold.*

Correctness *For all $\lambda, \beta, \nu_0, \nu_1$, letting* $(\mathsf{ot}_1, \mathsf{st}) = \mathsf{OT}_1(1^\lambda, \beta)$, $\mathsf{ot}_2 = \mathsf{OT}_2(1^\lambda, (\nu_0, \nu_1), \mathsf{ot}_1)$, $\nu' = \mathsf{OT}_3(1^\lambda, \mathsf{st}, \mathsf{ot}_2, \beta)$, *it holds that $\nu' = \nu_\beta$ with probability 1.*

Receiver Privacy *Consider the distribution $\mathcal{D}_\beta(\lambda)$ defined by running* $(\mathsf{ot}_1, \mathsf{st}) = \mathsf{OT}_1(1^\lambda, \beta)$ *and outputting* ot_1*. Then $\mathcal{D}_0, \mathcal{D}_1$ are computationally indistinguishable.*

Sender Privacy *There exists an (not necessarily efficient) extractor* OTExt *s.t. for any sequence of messages* $\mathsf{ot}_1 = \mathsf{ot}_1(\lambda)$ *and inputs (ν_0, ν_1), the distribution ensembles* $\mathsf{OT}_2(1^\lambda, (\nu_0, \nu_1), \mathsf{ot}_1)$ *and* $\mathsf{OT}_2(1^\lambda, (\nu_{\beta'}, \nu_{\beta'}), \mathsf{ot}_1)$, *where* $\beta' = \mathsf{OTExt}(\mathsf{ot}_1)$, *are statistically indistinguishable.*

2.5 Special Two-Round Oblivious Transfer

In this section, we introduce the security property and the syntax of the special two-round OT that we use in this work. This definition is taken from [IKSS21].

Syntax. Let $(\mathsf{OT}_1, \mathsf{OT}_1, \mathsf{OT}_3)$ be a two-round OT with the following syntax. OT_1 takes the security parameter 1^λ and the receiver's choice bit b and outputs the first round message ot_1 along with some secret state st. OT_2 takes ot_1 and the two sender inputs s_0 and s_1 as an input and outputs ot_2. OT_3 takes ot_2 and (b, st) as an input and outputs m_b. We require the OT protocol to satisfy the following properties:

Correctness: For every input b of the receiver and s_0, s_1 of the sender:

$$\Pr[\mathsf{OT}_3(\mathsf{ot}_2, (b, \mathsf{st})) = s_b] = 1$$

where $(\mathsf{ot}_1, \mathsf{st}) \leftarrow \mathsf{OT}_1(1^\lambda, b)$ and $\mathsf{ot}_2 \leftarrow \mathsf{OT}_2(\mathsf{ot}_1, (s_0, s_1))$.

Equivocal Receiver Security: There exists a special algorithm $\mathsf{Sim}^{eq}_{\mathsf{OT}}$ that on input 1^λ outputs $(\mathsf{ot}_1, \mathsf{st}_0, \mathsf{st}_1)$ such that for any $b \in \{0, 1\}$:

$$\{(\mathsf{ot}_1, \mathsf{st}_b) : (\mathsf{ot}_1, \mathsf{st}_0, \mathsf{st}_1) \leftarrow \mathsf{Sim}^{eq}_{\mathsf{OT}}(1^\lambda)\} \approx_c \{(\mathsf{ot}_1, \mathsf{st}) : (\mathsf{ot}_1, \mathsf{st}) \leftarrow \mathsf{OT}_1(1^\lambda, b)\}$$

Security in the No Corruption Setting: For any two bits b, b' and two sets of sender inputs (s_0, s_1) and (s'_0, s'_1), we have:

$$\{(\mathsf{ot}_1, \mathsf{ot}_2) : (\mathsf{ot}_1, \mathsf{st}) \leftarrow \mathsf{OT}_1(1^\lambda, b), \mathsf{ot}_2 \leftarrow \mathsf{OT}_2(\mathsf{ot}_1, (s_0, s_1))\} \approx_c \{(\mathsf{ot}_1, \mathsf{ot}_2) : (\mathsf{ot}_1, \mathsf{st}) \leftarrow \mathsf{OT}_1(1^\lambda, b'), \mathsf{ot}_2 \leftarrow \mathsf{OT}_2(\mathsf{ot}_1, (s'_0, s'_1))\}$$

Sender Privacy: For any input s_0, s_1 of the sender and any bit b and a string $r \in \{0, 1\}^*$:

$$\{b, r, \mathsf{ot}_1 := \mathsf{OT}_1(1^\lambda, b; r), \mathsf{OT}_2(\mathsf{ot}_1, (s_0, s_1))\}$$
$$\approx_c \{b, r, \mathsf{ot}_1 := \mathsf{OT}_1(1^\lambda, b; r), \mathsf{OT}_2(\mathsf{ot}_1, (s_b, s_b))\}$$

An OT that achieves the above notion is presented in [IKSS21, Appendix B], which can be instantiated, for example, using the OT protocols presented in [AIR01, Kal05, HK12, BD18].

2.6 Non-malleable Codes

Here, we recap the notion of non-malleable codes in the split-state model with one-many security and a special augmented non-malleability [AAG+16] property as they have been defined in [IKSS21].

Definition 2.7 (One-many augmented split-state non-malleable codes). *Fix any polynomials $\ell(\cdot), p(\cdot)$. An $\ell(\cdot)$-augmented non-malleable code with error $\epsilon(\cdot)$ for messages $m \in \{0, 1\}^{p(\lambda)}$ consists of algorithms* $\mathsf{NM} = (\mathsf{Code}, \mathsf{Decode})$ *where* $\mathsf{Code}(m) \to (L, R)$, *where $L \in \mathcal{L}$ and $R \in \mathcal{R}$ (we will assume that $\mathcal{L} = \mathcal{R}$) such that for every $m \in \{0, 1\}^{p(\lambda)}$,*

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

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

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

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.*

As already mentioned in [IKSS21], it was shown in [GSZ20, ADN$^+$19] that the CGL one-many non-malleable codes constructed in [CGL16] are also one-many *augmented*. Furthermore, as observed in [GJK15], to guarantee that a message obtained by decoding the tampered codewords with *left and right shares interchanged* is unrelated with the original message, any non-malleable code with symmetric decoding, i.e. where $\mathsf{Decode}(L, R) = \mathsf{Decode}(R, L)$ can be achieved by modifying any split-state code to attach "ℓ" to the left part of the codeword and "r" to the right part of the codeword. This results in the following imported lemma as stated in [IKSS21]:

Lemma 2.8 (Imported from [GJK15]). *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 2.7 with efficient encoding and decoding algorithms with code length $q(\lambda)$, rate $q(\lambda)^{-\Omega(1)}$ and error $2^{-q(\lambda)^{\Omega(1)}}$.*

2.7 Low-Depth Proofs

Now, we recap the notion of low-depth proof provided in [IKSS21]. Such a proof can be verified by a family of circuits in $\mathsf{NC1}$ and proves any computation performed by a polynomial sized circuits. Let R be an efficiently computable binary relation. Let L be the language consisting of statements in R, i.e. for which $R(x) = 1$.

Definition 2.9 (Low-Depth Non-Interactive Proofs). *A low-depth non-interactive proof with perfect completeness and soundness for a relation R consists fo 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 $\mathsf{NC1}$.*

A simple construction of a low-depth non-interactive proof has been introduced in [GGH$^+$13] and recapped in [IKSS21]. We refer to these works for further details.

2.8 MPC Definitions

Here, we provide a formal definition of secure multiparty computation verbatim taked from [IKSS21] which, in turn has been taken from [Ode09].

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

Defining Security We assume that the reader is familiar with standard simulation-based definitions of security multiparty computation in the standalone setting. We provide a self-contained definition for completeness and refer to [Ode09] for a more complete description. The security of a protocol (w.r.t. a functionality f) is defined by comparing the real-world execution of the protocol with an ideal-world evaluation of f by a trusted party. More concretely, it is required that for every adversary $\mathcal{A}$, which attacks the real execution of the protocol, there exists an adversary Sim, also referred to as the simulator, which can *achieve the same effect* in the ideal-world. We denote $\vec{x} = (x_1, \ldots, x_n)$.

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

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

The ideal execution - security with abort. In this model, an ideal execution for a function f proceeds as follows:

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

The interaction of Sim with the trusted party defines a random variable $\mathrm{Ideal}_{f,\mathsf{Sim}(z)}(\lambda, \vec{x})$ as above, and we let $\{\mathrm{Ideal}_{f,\mathsf{Sim}(z),I}(\lambda, \vec{x})\}_{\lambda \in \mathbb{N}, \vec{x}, z \in \{0,1\}^*}$. Having defined the real and the ideal worlds, we can now proceed to define the security notion.

Definition 2.10. *Let λ be the security parameter, f an n-party randomized functionality, and Π an n-party protocol for $n \in \mathbb{N}$. We say that Π t-securely computes f in the presence of malicious adversaries if for every PPT adversary $\mathcal{A}$ there exists a PPT adversary Sim such that for any $I \subset [n]$ with $|I| \leq t$ the following quantity is negligible:*

$$|\Pr[\mathrm{Real}_{\pi,\mathcal{A}(z),I}(\lambda, \vec{x}) = 1] - \Pr[\mathrm{Ideal}_{f,\mathcal{A}(z),I}(\lambda, \vec{x}) = 1]|,$$

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

Remark 2.11 (Security with Selective Abort). We can consider a slightly weaker definition of security where the ideal world adversary can instruct the trusted party to send aborts to a subset of the uncorrupted parties. For the rest of the uncorrupted parties, it instructs the trusted functionality to deliver their output. This weakened definition is called security with selective abort.

Remark 2.12 (Privacy with Knowledge of Outputs). Ishai et al. [IKP10] considered a further weakening of the security definition where the trusted party first delivers the output to the ideal world adversary which then provides an output to be delivered to all the honest parties. They called this security notion privacy with knowledge of outputs and showed a transformation from this notion to security with selective abort using unconditional MACs.

Security against Semi-Malicious Adversaries We take this definition almost verbatim from [AJL+12]. We consider a notion of a semi-malicious adversary that is stronger than the standard notion of semi-honest adversary and formalize security against semi-malicious adversaries. A semi-malicious adversary is modeled as an interactive Turing machine (ITM) which, in addition to the standard tape, has a special witness tape. In each round of the protocol, whenever the adversary produces a new protocol message msg on behalf of some party P_k, it must also write to its special witness tape some pair (x, r) of input x and randomness r that explains its behavior. More specifically, all of the protocol messages sent by the adversary on behalf of P_k up to that point, including the new message m, must exactly match the honest protocol specification for P_k when executed with input x and randomness r. Note that the witnesses given in different rounds need to be consistent. Also, we assume that the attacker is rushing and hence may choose the message m and the witness (x, r) in each round adaptively, after seeing the protocol messages of the honest parties in that round (and all prior rounds). Lastly, the adversary may also choose to abort the execution on behalf of P_k in any step of the interaction.

Definition 2.13. *We say that a protocol Π securely realizes f for semi-malicious adversaries if it satisfies Definition 2.10 when we only quantify over all semi-malicious adversaries $\mathcal{A}$.*

2.9 Outer Protocol

The outer protocol that is used in this work needs to achieve the same properties as the outer protocol used in [IKSS21] which we recap here verbatim.

Their outer protocol is a 2-round, n-client, m-server MPC protocol achieving privacy with knowledge of outputs (Remark 2.12) against a malicious, adaptive adversary corrupting up to $n-1$ clients and $t = (m-1)/3$ servers. Such a protocol was constructed in [IKP10] making black-box use of a pseudorandom generator (PRG). We set $m = 8\lambda n^2$. We now give details about the syntax of this protocol:

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

Remark 2.14. We require the functions computed by the servers to be information-theoretic and not involve any cryptographic operations. In the protocol of [IKP10], the server have to perform several PRG computations. To deal with this challenge, we delegate the computation of the PRG to each of the clients. Specifically, for every PRG computation to be done by each server, every client chooses a random seed and gives the output of the PRG on this seed to the server. Let $(\mathsf{seed}_i, \mathsf{PRG}(\mathsf{seed}_i))$ be the contribution from the i'th client where PRG has a sufficiently long stretch. The server sets $\mathsf{seed} = (\mathsf{seed}_1, \ldots, \mathsf{seed}_n)$ and defines a new $\mathsf{PRG}'(\mathsf{seed}) = \oplus_i \mathsf{PRG}(\mathsf{seed}_i)$. The seed and the PRG computation is sent as part of the first message from the client to the servers. The watchlist protocol is then used to ensure that the PRG computations done the honest servers are correct.

3 Bounded-Rewind Secure List 2-party Computation

In this section, we introduce the notion of bounded-rewind secure list 2-party computation. Let f be the function that the two parties P_0 and P_1 want to compute. $f : X^0 \times X^1 \to Y$ is a function that takes two inputs and returns two outputs (one for each party). We denote with $\mathcal{X}^d$ a PPT sampler for X^d for each $d \in \{0, 1\}$.

The notion of list 2PC differs from the standard notion of 2PC as follows. Let us say that the party P_0 is corrupted and P_1 is honest. In the ideal world experiment, the adversary sends its input $x^0 \in \mathcal{X}^0$ to the ideal functionality, which we denote with $\mathcal{F}^{\mathcal{X}^1}$, together with an integer k. The ideal functionality does not wait to receive the input from the honest party, instead, it samples k inputs uniformly at random from the input domain of the honest party $x_1^1, \dots, x_k^1 \leftarrow \mathcal{X}^1$, and, for each $j \in [k]$ computes $(\mathsf{out}_j^0, \mathsf{out}_j^1) \leftarrow f(x^0, x_j^1)$. Then the values $(\mathsf{out}_1^0, \dots, \mathsf{out}_k^0)$ are given to the adversary. The adversary now sends an index $i \in [k]$ to the ideal functionality, which then sends $(x_i^1, \mathsf{out}_i^1)$ to the ideal-world honest party.

In our work, we focus on constructing and studying 3-round list 2PC protocols, where in each round both parties speak at the same time. We study our notion in a stronger adversarial setting where the adversary can ask the honest party to receive multiple third rounds for multiple second rounds. A protocol that is proven secure in this setting is usually said to be B-rewind secure, which means that the adversary can send $B+1$ second rounds and receive $B+1$ third rounds (one for each second round). We propose the formal definition of the real and the ideal world in Figure 3.2, where the party P^d is assumed to be corrupted with $d \in \{0,1\}$. To make a protocol that satisfies our definition usable as a sub-routine of other protocols, we need to make sure that the distinguisher has somehow access to the inputs of the honest parties. In the ideal world, we do that by explicitly giving this input to the adversary together with the second round of the protocol. In the ideal world experiment, we do something similar, by allowing sending to the adversary the input of the honest parties upon request of the simulator. More precisely, if the simulator sends a message to the adversary (π, j), then the real-world experiment would forward to the adversary the message (π, x_j^{1-d}). This guarantees that the distinguisher will get access to the honest party's input, while the simulator does not (this is exactly what makes our definition non-trivial to realize). We refer to Figure 3.2 for the formal specification of real and ideal world and state the following:

Definition 3.1 (B-rewind secure list 2PC). *We say that a protocol Π is a B-rewind secure List 2PC if for every malicious PPT P_{1-d}, with $d \in \{0,1\}$, there exists a (expected) PPT simulator $\mathsf{Sim} = \{\mathsf{Sim}_1, \mathsf{Sim}_2\}$ such that*

$$\{\mathsf{Real}_{\mathcal{A},\Pi}(1^\lambda, 1^B)\}_{\lambda,B} \approx_c \{\mathsf{Ideal}_{\mathsf{Sim},\mathcal{F}^{\mathcal{X}^d}}(1^\lambda, 1^B)\}_{\lambda,B},$$

where $\lambda, B \in \mathbb{N}$.

Sometimes aborting adversaries. As previously mentioned, in this paper we consider the notion of sometimes aborting adversaries. This notion is the same as the list simulation notion, but it requires the list simulator to provide a simulated transcript with overwhelming probability, conditioned on the adversary providing an accepting transcript in the main thread. We still provide no security requirements (unless otherwise specified), in the case the adversary aborts with overwhelming probability, or it does not provide an accepting transcript in the main thread.

Figure 3.1: Real and ideal world

$\mathsf{Real}_{\mathcal{A},\Pi}(1^\lambda, 1^B)$:

1. Sample $x_d \leftarrow \mathcal{X}^d$, and compute the first round π_1^d of Π as the honest P_d would do on input x_d and the randomness $r \leftarrow \{0,1\}^\lambda$ and send π_1^d to $\mathcal{A}$.
2. Upon receiving π_1^{1-d} from $\mathcal{A}$, compute the second round π_2^d of Π as the honest party P_d on input (r, x_d) would do and send (π_2^d, x_d) to $\mathcal{A}$.
3. Upon receiving $\{\pi_{2,i}^{1-d}\}_{i\in[B]}$ from $\mathcal{A}$, for each $i \in [B]$, compute $\pi_{3,i}^d$ as the honest party P_d would do having on input $(\pi_1^{1-d}, \pi_{2,i}^{1-d}, r, x_d)$.
4. Send $\{\pi_{3,i}^d\}_{i\in[B]}$ to $\mathcal{A}$.
5. Upon receiving π_3^{1-d}, compute the output out as P_d would do, and return the view of $\mathcal{A}$.

$\mathsf{Ideal}_{\mathsf{Sim},\mathcal{F}^{\mathcal{X}^d}}(1^\lambda, 1^B)$:

1. $(x^{1-d}, 1^k, z) \leftarrow \mathsf{Sim}_1^{\mathcal{A}}(1^\lambda, 1^B)$
2. Send x^{1-d} to the ideal functionality $\mathcal{F}^{\mathcal{X}^d}$ which computes $\{\mathsf{out}_j^0, \mathsf{out}_j^1\}_{j\in[k]}$ as described using the sampled inputs $\{x_j^d\}_{j\in[k]}$.
3. Whenever $\mathsf{Sim}_2^{\mathcal{A}}(\mathsf{out}_1^{1-d}, \dots, \mathsf{out}_k^{1-d}, z)$ queries $\mathcal{A}$ with a message (π, j), replace the query with (π, x_j^d) and forward the pair to $\mathcal{A}$.
4. $(i, \text{View}) \leftarrow \mathsf{Sim}_2^{\mathcal{A}}(\mathsf{out}_1^{1-d}, \dots, \mathsf{out}_k^{1-d}, z)$ with $i \in [k] \cup \{\perp\}$.
5. Send i to the ideal functionality, or abort if $i = \perp$ to instruct the honest party to abort, and return View.

List Oblivious Transfer. One of the main tools we construct is a protocol that realizes the OT functionality $\mathcal{F}_{\mathsf{OT}}$ with list simulation against corrupted sometimes aborting receivers. Then we show how to use such an OT to construct a 2PC protocol (where only one party gets the output) that is list simulatable against a corrupted sender, and list simulatable against sometimes aborting receivers. We then use the 2PC protocol to obtain a non-malleable OT protocol, which retains the same security as the 2PC protocol.

3.1 List Multiparty Computation

Following what we have done in Section 3 we present the notion of *List Multiparty Computation.* Let f be the function that n parties $P_1, \dots, P_n$ want to compute: $f : X^0 \times \cdots \times X^n \to Y^1 \times \cdots \times Y^n$. We denote with $\mathcal{X}^i$ a PPT sampler for X^i for each $i \in [n]$. The notion of list MPC differs from the standard notion of MPC as follows. Let $M \subseteq [n]$ denote the indices of the corrupted parties and $H := [n] \setminus M$. In the ideal world experiment, the adversary sends its inputs $\{x^i \in \mathcal{X}^i\}_{i \in M}$ to the ideal functionality, which we denote with $\mathcal{F}^{\{\mathcal{X}^i\}_{i \in H}}$, together with an integer k. The ideal functionality does not wait to receive the input from the honest parties, instead, it samples k inputs uniformly at random from the input domains of the honest parties $\{x^i_1 \leftarrow \mathcal{X}^i\}_{i \in H}, \dots, \{x^i_k \leftarrow \mathcal{X}^i\}_{i \in H}$, and, for each $j \in [k]$ computes $(\mathsf{out}^1_j, \dots, \mathsf{out}^n_j) \leftarrow f(\{x^i_j\}_{i \in H}, \{x^i_j\}_{i \in M})$. Then the values $\{\mathsf{out}^i_1, \dots, \mathsf{out}^i_k\}_{i \in M}$ are given to the adversary. The adversary now sends an index $j \in [k]$ to the ideal functionality, which then sends $(x^i_j, \mathsf{out}^i_j)$ to the ideal-world honest party P_i for each $i \in H$.

We propose the formal definition of the real and the ideal world in Figure 3.2, and below we state the formal security definition.

Definition 3.2 (List MPC). *We say that a protocol Π is a secure List MPC protocol if for every malicious PPT adversary $\mathcal{A}$ corrupting an arbitrary set of parties in indices $M \subseteq [n]$ (the indices of the honest parties are denoted with $H := [n] \setminus M$), there exists a (expected) PPT simulator $\mathsf{Sim} = \{\mathsf{Sim}_1, \mathsf{Sim}_2\}$ such that*

$$\{\mathsf{Real}_{\mathcal{A},\Pi}(1^\lambda, M, H)\}_\lambda \approx_c \{\mathsf{Ideal}_{\mathsf{Sim},\mathcal{F}^{\{\mathcal{X}^i\}_{i \in [H]}}}(1^\lambda, M, H)\}_\lambda,$$

where $\lambda \in \mathbb{N}$.

Figure 3.2: Real and ideal world

$\mathsf{Real}_{\mathcal{A},\Pi}(1^\lambda, M, H)$:

1. For each $i \in H$ sample $x_i \leftarrow \mathcal{X}^i$, and compute the first round π^i_1 of Π as the honest P_i would do on input x_i and the randomness $r_i \leftarrow \{0,1\}^\lambda$ and send π^i_1 to $\mathcal{A}$.
2. Upon receiving $\{\pi^i_1\}_{i \in M}$ from $\mathcal{A}$, for each $i \in H$ compute the second round π^i_2 of Π as the honest party P_i on input (r, x_i) would do and send (π^i_2, x_i) to $\mathcal{A}$.
3. Upon receiving $\{\pi^i_2\}_{i \in M}$ from $\mathcal{A}$, for each $i \in H$ compute π^i_3 as the honest party P_i would do having on input $(\pi^i_1, \pi^i_2, r_i, x_i)$ and send π^i_3 to $\mathcal{A}$.
4. Upon receiving $\{\pi^i_3\}_{i \in M}$, compute the output out_i as P_i would do, and return the view of $\mathcal{A}$.

$\mathsf{Ideal}_{\mathsf{Sim},\mathcal{F}^{\{\mathcal{X}^i\}_{i \in M}}}(1^\lambda, M, H)$:

1. $(\{x^i\}_{i \in M}, 1^k, z) \leftarrow \mathsf{Sim}^{\mathcal{A}}_1(1^\lambda)$
2. Send $\{x^i\}_{i \in M}$ to the ideal functionality $\mathcal{F}^{\{\mathcal{X}^i\}_{i \in M}}$ which computes $\{\mathsf{out}^i_j\}_{i \in M, j \in [k]}$ as described using the sampled inputs $\{x^i_j\}_{j \in [k]}$.
3. Whenever $\mathsf{Sim}^{\mathcal{A}}_2(\{\mathsf{out}^i_j\}_{i \in M, j \in [k]}, z)$ queries $\mathcal{A}$ with a message $(\{\pi_i\}_{i \in H}, j)$, replace the query with $(\{\pi_i, x^i_j\}_{i \in H})$ and forward the pair to $\mathcal{A}$.
4. $(i, \text{View}) \leftarrow \mathsf{Sim}^{\mathcal{A}}_2(\{\mathsf{out}^i_j\}_{i \in M, j \in [k]}, z)$ with $i \in [k] \cup \{\bot\}$.
5. Send i to the ideal functionality, or abort if $i = \bot$ to instruct the honest party to abort, and return View.

3.2 List 2PC with delayed-input function selection

Definition 3.3 (Adapted from [IKSS21]). *Let $\Pi = ((\Pi^S_1, \Pi^R_1), (\Pi^S_2, \Pi^R_2), (\Pi^S_3, \Pi^R_3), (\Pi^S_4, \Pi^R_4), \mathsf{out})$ be a 3-round protocol (in the simultaneous message model) between a receiver R and a sender S with the receiver computing the output at the end of the third round. We say that Π is* 1*-rewinding secure list-simulatable with delayed function selection for NC^1 circuits if it satisfies the following:*

Delayed function Selection. *The first message functions Π^S_1, Π^R_1 and the second message function of the sender Π^S_2 take as input the size of the function $f \in \mathsf{NC}^1$ to be securely computed and are otherwise, independent of the function description. The second round message from R contains the explicit description of the function f to be computed.*

1-**Rewind receiver security.** *For every malicious PPT adversary $\mathcal{A}$ that corrupts the sender, there exists an expected polynomial (black-box) simulator $\mathsf{Sim}_S = (\mathsf{Sim}^1_S, \mathsf{Sim}^2_S)$ such that for all choices of honest receiver input x_R and the function $f \in \mathsf{NC}^1$ the joint distribution of the view of $\mathcal{A}$ and R's output in the real execution is computationally indistinguishable from the output of the ideal experiment described in Figure 3.3.*

Figure 3.3: 1-*Rewind receiver security*

$\mathsf{Real}_{\mathcal{A},\Pi}(x_R)$:

1. *Initialize $\mathcal{A}$ with a uniform random tape s.*
2. *Compute $\pi^R_1 \leftarrow \Pi^R_1(x_R)$ and send it to $\mathcal{A}$.*
3. *Upon receiving π^S_1, use it to run Π^R_2 thus obtaining π^R_2, and send it to $\mathcal{A}$.*
4. *Upon receiving $(\pi^S_2[0], \pi^S_2[1])$ from $\mathcal{A}$ use Π^R_2 to compute the third round $\pi^R_3[b]$ for each $b \in \{0,1\}$ and send $(\pi^R_3[0], \pi^R_3[1])$ to $\mathcal{A}$.*
5. *Output whatever $\mathcal{A}$ outputs.*

$\mathsf{Ideal}_{\mathsf{Sim}_S,\mathcal{F}}(x_R)$:

1. *Initialize $\mathcal{A}$ with a uniform random tape s.*
2. $(x^S, z) \leftarrow \mathsf{Sim}^1_S(1^\lambda)$.
3. *Send (x^S) to the ideal functionality $\mathcal{F}$ which returns* $\mathsf{out} \leftarrow f(x^S, x^R)$.
4. $(\mathsf{abort}, \mathrm{View}) \leftarrow \mathsf{Sim}^2_S(\mathsf{out}, z)$.
5. *If* $\mathsf{abort} = 1$ *then return* abort *to the ideal functionality, else return* $\mathsf{continue}$.
6. *Return* View.

1-**Rewind sender security.** *As in the definition of list-simulation security, ideal and real world are parametrized by a distribution $\mathcal{X}_S$. The input of the sender is sampled from $\mathcal{X}_S$ in the real-world experiment and used to execute Π. The ideal-world experiment is exactly the same as the one of Definition 3.2, with the difference that the ideal functionality $\mathcal{F}^{\mathcal{X}_S}$, in addition to the input of the corrupted receiver x_R and 1^k, receives two functions (f_0, f_1). The ideal functionality then, for each $i \in [k]$ sample $x_S \leftarrow \mathcal{X}_S$ and compute $\mathsf{out}^0_i \leftarrow f_0(x_S, x_R)$, $\mathsf{out}^1_i \leftarrow f_1(x_S, x_R)$. Then the ideal functionality returns $\{\mathsf{out}^0_i, \mathsf{out}^1_i\}_{i\in[k]}$. We provide the formal description of the ideal and real-world in Figure 3.4, and we say that a protocol Π is* 1*-rewind sender secure if for every malicious adversary $\mathcal{A}$ corrupting the receiver which is non-aborting in the last round, there exists an expected polynomial time simulator $\mathsf{Sim}_R = (\mathsf{Sim}^1_R, \mathsf{Sim}^2_R)$ such that, given $\mathcal{X}_S$ we have $\{\mathsf{Real}_{\mathcal{A},\Pi}(1^\lambda)\} \approx \{\mathsf{Ideal}_{\mathsf{Sim}_R,\mathcal{F}^{\mathcal{X}_S}}(1^\lambda)\}$.*

Figure 3.4: 1-*Rewind sender security.*

$\mathsf{Real}_{\mathcal{A},\Pi}(1^\lambda)$:

1. *Initialize $\mathcal{A}$ with a uniform random tape s, and sample $x_S \leftarrow \mathcal{X}_S$.*
2. *Compute $\pi^S_1 \leftarrow \Pi^S_1(x_S)$ and send it to $\mathcal{A}$.*
3. *Upon receiving π^R_1, use it to run Π^S_2 thus obtaining π^S_2, and send (π^S_2, x_S) to $\mathcal{A}$.*
4. *Upon receiving $(f_0, f_1, \pi^R_2[0], \pi^R_2[1])$ from $\mathcal{A}$ use $(f_b, \pi^R_2[b])$ to compute the third round $\pi^S_3[b]$ for each $b \in \{0,1\}$ and send $\pi^S_3[0], \pi^S_3[1]$ to $\mathcal{A}$.*
5. *Output whatever $\mathcal{A}$ outputs.*

$\mathsf{Ideal}_{\mathsf{Sim}_R,\mathcal{F}^{\mathcal{X}_S}}(1^\lambda)$:

1. *Initialize $\mathcal{A}$ with a uniform random tape s.*
2. $(x_R, f_0, f_1, 1^k, z) \leftarrow \mathsf{Sim}^1_R(1^\lambda, 1^B)$
3. *Send (x_R, f_0, f_1) to the ideal functionality $\mathcal{F}^{\mathcal{X}_S}$ which returns $\{\mathsf{out}^0_j, \mathsf{out}^1_j\}_{j\in[k]}$ computed as described using the sampled inputs $\{x_{S,j}\}_{j\in k}$.*
4. *Whenever Sim^2_R queries $\mathcal{A}$ with a message (π^R, j), replace the query with $(\pi^R, x_{S,j})$.*
5. $(i, \mathrm{View}) \leftarrow \mathsf{Sim}^2_R((\mathsf{out}^0_1, \mathsf{out}^1_1), \ldots, (\mathsf{out}^0_k, \mathsf{out}^0_k), z)$ *with $i \in [k] \cup \{\bot\}$.*
6. *Send i to the ideal functionality, or* abort *if $i = \bot$ to instruct the honest party to abort, and return* View.

Definition 3.4. *Let $\Pi = ((\Pi^S_1, \Pi^R_1), (\Pi^S_2, \Pi^R_2), (\Pi^S_3, \Pi^R_3), (\Pi^S_4, \Pi^R_4), \mathsf{out})$ be a 3-round protocol (in the simultaneous message model) between a receiver R and a sender S with the receiver computing the output at the end of the third round. We say that Π is* 1*-rewinding secure ℓ-senders list-simulatable with delayed function selection for* NC^1 *circuits if it satisfies the properties of delayed function selection and 1-rewind receiver security as defined in Definition 3.3 and moreover it enjoys the following property:*

1-**Rewind ℓ-senders security.** *In this definition we consider a real-world experiment where $\ell \in \mathrm{poly}(\lambda)$ senders $S_1, \ldots, S_\ell$ interact with ℓ malicious receivers $R^*_1, \ldots, R^*_\ell$ in an execution of Π; specifically the S_j interacts with R^*_j for $j \in [\ell]$. The ideal and real world are parametrized by distributions $\{\mathcal{X}^j_S\}_{j\in[\ell]}$. The input of the j-th sender S_j is sampled from $\mathcal{X}^j_S$ in the real-world experiment and used to execute Π. The ideal-world experiment is exactly the same as the one of Definition 3.3, with the difference that the ideal* *functionality $\mathcal{F}^{\{\mathcal{X}_S^j\}_{j\in[\ell]}}$ computes the output for ℓ executions of Π. We provide the formal description of the ideal and real-world in Figure 3.5, and we say that a protocol Π is 1-rewind ℓ-senders secure if for every malicious non-uniform adversary $\mathcal{A}$ corrupting the ℓ receivers which are non-aborting in the last round, there exists an expected polynomial time simulator $\mathsf{Sim}_S = (\mathsf{Sim}_S^1, \mathsf{Sim}_S^2)$ such that, given $\{\mathcal{X}_S^j\}_{j\in[\ell]}$ we have $\{\mathsf{Real}_{\mathcal{A},\Pi}(1^\lambda)\} \approx \{\mathsf{Ideal}_{\mathsf{Sim}_S,\mathcal{F}^{\{\mathcal{X}_S^j\}_{j\in[\ell]}}}(1^\lambda)\}$.*

Figure 3.5: 1-*Rewind ℓ-senders security.*

$\mathsf{Real}_{\mathcal{A},\Pi}(1^\lambda)$:

1. *Initialize $\mathcal{A}$ with a uniform random tape s, and sample $x_{S_j} \leftarrow \mathcal{X}_S^j$, for $j \in [\ell]$.*
2. *Compute $\pi_1^{S_j} \leftarrow \Pi_1^{S_j}(x_{S_j})$ and send it to $\mathcal{A}$, for $j \in [\ell]$.*
3. *Upon receiving $\pi_1^{R_j}$, use it to run $\Pi_2^{S_j}$ thus obtaining $\pi_2^{S_j}$, and send $(\pi_2^{S_j}, x_{S_j})$ to $\mathcal{A}$, for $j \in [\ell]$.*
4. *Upon receiving $(f_0^j, f_1^j, \pi_2^{R_j}[0], \pi_2^{R_j}[1])$ from $\mathcal{A}$ use $(f_b^j, \pi_2^{R_j}[b])$ to compute the third round $\pi_3^{S_j}[b]$ for each $b \in \{0,1\}$ and send $\pi_3^{S_j}[0], \pi_3^{S_j}[1]$ to $\mathcal{A}$, for $j \in [\ell]$.*
5. *Output whatever $\mathcal{A}$ outputs.*

$\mathsf{Ideal}_{\mathsf{Sim}_S,\mathcal{F}^{\{\mathcal{X}_S^j\}_{j\in[\ell]}}}(1^\lambda)$:

1. *Initialize $\mathcal{A}$ with a uniform random tape s.*
2. $\{(x_{R_j}, f_0^j, f_1^j, 1^\kappa, z)\}_{j\in[\ell]} \leftarrow \mathsf{Sim}_S^1(1^\lambda, 1^B)$
3. *Send $\{(x_{R_j}, f_0^j, f_1^j)\}_{j\in[\ell]}$ to the ideal functionality $\mathcal{F}^{\{\mathcal{X}_S^j\}_{j\in[\ell]}}$ which returns $\{\mathsf{out}_{j,k}^0, \mathsf{out}_{j,k}^1\}_{j\in[\ell],k\in[\kappa]}$ computed as described using the sampled inputs $\{x_{S_j,k}\}_{j\in[\ell],k\in[\kappa]}$.*
4. *Whenever Sim_S^2 queries $\mathcal{A}$ with a message (π_{R_j}, k), replace the query with $(\pi R_j, x_{S_j,k}^d)$.*
5. $(i, \mathrm{View}) \leftarrow \mathsf{Sim}_S^2((\{\mathsf{out}_{j,1}^0, \mathsf{out}_{j,1}^1), \ldots, (\mathsf{out}_{j,\kappa}^0, \mathsf{out}_{j,\kappa}^0\}_{j\in[\ell]}), z)$ *with $i \in [\kappa] \cup \{\perp\}$.*
6. *Send i to the ideal functionality, or* abort *if $i = \perp$ to instruct the honest party to abort, and return* View.

3.3 Bounded Rewind Receiver Private OT

Definition 3.5 (Bounded Rewind Receiver Private OT). *Let $\mathsf{OT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3, \mathsf{OT}_4)$ be an OT protocol sender list simulatable, then we say that OT is special B-rewindable secure against malicious senders with B rewinds if the output distributions of the adversary in the experiments E^0 and E^1 (where E^σ is defined in Figure 1) are computationally indistinguishable for any $k \in [B]$ and all $\{b^0[j], b^1[j]\}_{j\in[B]}$ with $b^\sigma[j] \in \{0,1\}^\lambda$ for all $j \in [B]$ and $\sigma \in \{0,1\}$ and with $b^\sigma[j-1] = b^\sigma[j]$.*

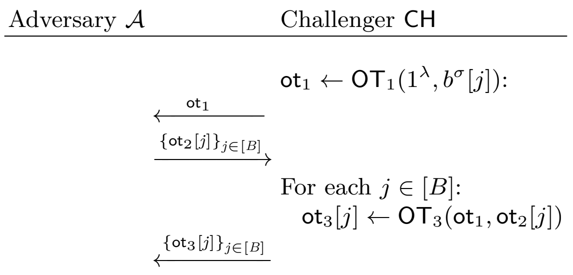
Adversary $\mathcal{A}$
Challenger CH
$\mathsf{ot}_1 \leftarrow \mathsf{OT}_1(1^\lambda, b^\sigma[j])$:
ot_1
$\{\mathsf{ot}_2[j]\}_{j\in[B]}$
For each $j \in [B]$:
$\mathsf{ot}_3[j] \leftarrow \mathsf{OT}_3(\mathsf{ot}_1, \mathsf{ot}_2[j])$
$\{\mathsf{ot}_3[j]\}_{j\in[B]}$

Fig. 1: The experiment E^σ.

We note that this definition is similar to the one proposed in [CCG+20] except for the fact that we require the adversary to pick the same inputs in the B rewinds. We call a protocol receiver private if it is 1-rewind secure. Finally we notice that this notion compose under parallel repetition.

4 Sender List Simulatable OT

In this section, we construct a three-round OT protocol that is: sender side list simulatable against a sometimes aborting adversaries, and sender private otherwise.

As an intermediate step, we propose a 3-round oblivious transfer in the alternating message model that is not rewind-secure. This protocol consists of the following algorithms $\mathsf{OT}' = (\mathsf{OT}'_1, \mathsf{OT}'_2, \mathsf{OT}'_3, (\mathsf{OT}'^S_4, \mathsf{OT}'^R_4))$ where OT'^R_4 is the procedure used to compute the output of the OT protocol by the receiver and OT'^S_4 is the procedure run by the sender to decide whether to abort or not. We present the formal description of the protocol in Figure 4.1, and refer the reader to the introductory part of the paper for an informal description of the scheme. The only tool that we need for this construction is a maliciously private two-round OT protocol denoted as $\mathsf{OT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ (Definition 2.6), where OT_3 is the procedure used by the receiver to compute the output of the OT protocol.

Figure 4.1: 3-round sender list simulatable and receiver private OT OT'

Initialization: The receiver uses as its input a bit b and the sender has input s_0, s_1. The parties also receive a common parameter $m = \text{poly}(\lambda)$ as an input.

Round 1 (Receiver).

1. Sample $b_i \leftarrow \{0,1\}$ and compute $d_i = b_i \oplus b$ for all $i \in [m]$.
2. Generate $\mathsf{ot}_{1,i} := \mathsf{OT}_1(1^\lambda, b_i; r_i)$ and $\mathsf{ot}'_{1,i} := \mathsf{OT}_1(1^\lambda, d_i; r'_i)$ with $r_i, r'_i \leftarrow \{0,1\}^\lambda$ for all $i \in [m]$.
3. Send $\{\mathsf{ot}_{1,i}, \mathsf{ot}'_{1,i}\}_{i\in[m]}$ to S.

Round 2 (Sender).

1. Sample $\mathsf{k}^i_c, s_{c,j} \leftarrow \{0,1\}^\lambda$ for all $i \in [m], j \in [m-1], c \in \{0,1\}$. Set $s_{c,m} := s_c \oplus s_{c,1} \oplus \cdots \oplus s_{c,m-1}$ for both $c \in \{0,1\}$.
2. Compute $\mathsf{ct}^{0,i}_c := \mathsf{k}^i_c \oplus s_{c,i}$ and $\mathsf{ct}^{1,i}_c := \mathsf{k}^i_{1\oplus c} \oplus s_{c,i}$ for all $i \in [m], c \in \{0,1\}$.
3. Generate $\mathsf{ot}_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}_{1,i}, (\mathsf{k}^i_0, \mathsf{k}^i_1))$ and $\mathsf{ot}'_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}'_{1,i}, ((\mathsf{ct}^{0,i}_0, \mathsf{ct}^{0,i}_1), (\mathsf{ct}^{1,i}_0, \mathsf{ct}^{1,i}_1)))$ for all $i \in [m]$.
4. Sample $\mathcal{I} \leftarrow \{0,1\}^m$.
5. Send $\{\mathsf{ot}_{2,i}, \mathsf{ot}'_{2,i}\}_{i\in[m]}$ and $\mathcal{I}$ to R.

Round 3 (Receiver).

1. Send $\{b_i, r_i\}_{i\in[m]:\mathcal{I}_i=1}$ and $\{d_i, r'_i\}_{i\in[m]:\mathcal{I}_i=0}$ to S.

Offline computation.

Sender:

1. Check that $\mathsf{ot}_{1,i} := \mathsf{OT}_1(1^\lambda, b_i; r_i)$ for all $i \in [m]$ where $\mathcal{I}_i = 1$ and $\mathsf{ot}'_{1,i} := \mathsf{OT}_1(1^\lambda, d_i; r'_i)$ for all $i \in [m]$ where $\mathcal{I}_i = 0$.
2. If the previous check does not succeed output $\perp$.

Receiver:

1. Obtain $\mathsf{k}^i_{b_i} := \mathsf{OT}_3(\mathsf{ot}_{1,i}, \mathsf{ot}_{2,i})$ and $(\mathsf{ct}^{d_i,i}_0, \mathsf{ct}^{d_i,i}_1) := \mathsf{OT}_3(\mathsf{ot}_{1,i}, \mathsf{ot}_{2,i})$ for all $i \in [m]$.
2. Compute $s'_{b,i} := \mathsf{k}^i_{b_i} \oplus \mathsf{ct}^{d_i,i}_b$ for all $i \in [m]$ and output $s'_b := \bigoplus_{i\in[m]} s'_{b,i}$.

Before we argue the security of our OT protocol OT', we prove its correctness. To argue the correctness, we need to show that $s'_b = s_b$ if the sender and the receiver followed the protocol. If both parties are honest, it follows that:

$$\begin{aligned} s'_b &= \bigoplus_{i\in[m]} s'_{b,i} = \bigoplus_{i\in[m]} (\mathsf{k}^i_{b_i} \oplus \mathsf{ct}^{d_i,i}_b) = \bigoplus_{i\in[m]} (\mathsf{k}^i_{b_i} \oplus \mathsf{k}^i_{d_i\oplus b} \oplus s_{b,i}) \\ &= \bigoplus_{i\in[m]} (\mathsf{k}^i_{b_i} \oplus \mathsf{k}^i_{b_i\oplus b\oplus b} \oplus s_{b,i}) = \bigoplus_{i\in[m]} (\mathsf{k}^i_{b_i} \oplus \mathsf{k}^i_{b_i} \oplus s_{b,i}) = \bigoplus_{i\in[m]} s_{b,i} = s_b \end{aligned}$$

Theorem 4.1. *Let X be a high min-entropy random variable defined by a probability distribution $\mathcal{D}$ and let* OT *be a two round maliciously private OT protocol, then the OT protocol* OT' *described in Figure 4.1* *is a three-round sender list simulatable OT against sometimes aborting receivers for the functionality $\mathcal{F}_{\mathsf{OT}}^{\mathcal{D}}$ (according to Definition 3.2) and sender private OT otherwise (according to Definition 2.6). Moreover, OT' is receiver private (according to Definition 2.6). OT' makes black-box use of OT.*

We split the theorem into three lemmas, the first two focusing on the simulatability (Lemma 4.2) and privacy (Lemma 4.9) against a malicious receiver and the third focusing on the privacy of the receiver against a malicious sender (Lemma 4.14). The first two lemmas are stated and proven in Sections 4.1 and 4.2, whereas the third is proven in Section 4.3.

4.1 List Simulatability against a malicious Receiver

In this section, we prove the list-simulatability against a malicious receiver.

Lemma 4.2. *Let X be a high min-entropy random variable defined by a probability distribution $\mathcal{D}$ and let OT be a two-round maliciously private OT protocol, then the OT protocol OT' described in Figure 4.1 is a three-round sender list simulatable OT against a non-aborting receiver for the functionality $\mathcal{F}_{\mathsf{OT}}^{\mathcal{D}}$ (accordingly to Definition 3.2).*

Proof. We prove that the OT OT' described in Figure 4.1 is sender list simulatable by showing that a malicious (non-aborting) receiver is not able to distinguish between a real execution of the protocol and an ideal execution using the simulator $\mathsf{Sim}' = (\mathsf{Sim}'^1, \mathsf{Sim}'^2)$ described in Figure 4.2. The proof that this simulator runs in expected polynomial-time and succeeds with overwhelming probability can be found in Claim 4.3 & Claim 4.4.

We prove the indistinguishability between the real world and the ideal world using a sequence of hybrids that we describe below:

Hybrid H_0: This hybrid corresponds to the real world.

Hybrid H_1: This hybrid is almost identical to the previous hybrid with the only difference that the input bit b of the receiver is extracted. In order to do so the hybrid follows the steps (1), (2) and (3) of Sim' described in Figure 4.2, the hybrid outputs the view of R^* in the first thread. The indistinguishability of this and the previous hybrid follows with similar arguments to the one described in Claim 4.3 & (first part of claim) Claim 4.4.

Hybrid H_1': In this hybrid, we proceed as in the previous hybrid except that, after the extraction of the input, the hybrid proceeds as explained in steps (4) and (6) of Sim' described in Figure 4.2, with the difference that the second round is computed honestly (as in steps (2) and (3)). The hybrid outputs the same output as Sim'. The indistinguishability of this and the previous hybrid follows with similar arguments to the one described in Claim 4.3 & (first two parts of) Claim 4.4.

Hybrid H_2^x: In this hybrid, we proceed as in the previous hybrid but in step (x).g the message ot_{2,j^*} is generated as described in step (x).g of Figure 4.2 (where x is used to denote the 4th or the 6th stage of the simulator, namely $x = 4$ or $x = 6$). In more detail the hybrid only samples one of the keys $\mathsf{k}_0^{j^*}$ and $\mathsf{k}_1^{j^*}$ that are used for the generation of the OT messages ot_{2,j^*} randomly and sets the other to the all-zero string. Formally, $\mathsf{k}_{b_{j^*}}^{j^*} \leftarrow \{0,1\}^\lambda$ and $\mathsf{k}_{1\oplus b_{j^*}}^{j^*} := 0^\lambda$. For the remaining indices $i \in [m] \setminus \{j^*\}$ both of the keys remain randomly sampled $\mathsf{k}_0^i, \mathsf{k}_1^i \leftarrow \{0,1\}^\lambda$. The indistinguishability of this and the previous hybrid follows from the sender privacy of the underlying OT protocol OT, which we formally prove in Claim 4.5.

Hybrid H_3^x: In this hybrid, we proceed as in the previous hybrid except that in step (x).2.e the ciphertexts $\mathsf{ct}_0^{d_{j^*},j^*}, \mathsf{ct}_1^{d_{j^*},j^*}$ are generated correctly[12] while the ciphertexts for the other slot $\mathsf{ct}_0^{d_{j^*}\oplus 1,j^*}, \mathsf{ct}_1^{d_{j^*}\oplus 1,j^*}$ are set to the all-zero string, i.e. $\mathsf{ct}_0^{d_{j^*}\oplus 1,j^*} := 0^\lambda, \mathsf{ct}_1^{d_{j^*}\oplus 1,j^*} := 0^\lambda$. For the remaining indices $i \in [m] \setminus \{j^*\}$ all of the ciphertexts remain correctly generated, i.e. $\mathsf{ct}_c^{0,i} := \mathsf{k}_c^i \oplus s_{c,i}$ and $\mathsf{ct}_c^{1,i} := \mathsf{k}_{1\oplus c}^i \oplus s_{c,i}$ for $c \in \{0,1\}$. The indistinguishability this and the previous hybrid follows, similar to the previous two hybrids, i.e. from the sender privacy of the underlying OT protocol OT, which we formally prove in Claim 4.6.

[12] Here, for the generation of the ciphertext $\mathsf{ct}_{1\oplus b}^{d_{j^*},j^*}$, we still use a random key $\mathsf{k}'^{j^*}_{1\oplus b_{j^*}}$ for the encryption and not the all-zero key $\mathsf{k}^{j^*}_{1\oplus b_{j^*}}$.

Hybrid H_4^x: In this hybrid, we proceed as in the previous hybrid except that in step (x).2.e the ciphertext $\mathsf{ct}_{1\oplus b}^{d_{j^*},j^*} := \mathsf{k}'^{j^*}_{1\oplus b_{j^*}} \oplus s_{1\oplus b,j^*}$ is changed to a randomly sampled ciphertext $\mathsf{ct}_{1\oplus b}^{d_{j^*},j^*} \leftarrow \{0,1\}^\lambda$. For the remaining indices $i \in [m] \setminus \{j^*\}$ all the ciphertexts remain generated as described in the protocol. Since, due to the previous hybrid, the key $\mathsf{k}^{j^*}_{1\oplus b_{j^*}}/\mathsf{k}'^{j^*}_{1\oplus b_{j^*}}$ is not part of the execution anymore, we can rely on the perfect security of the one-time pad to show that this and the previous hybrid are perfectly indistinguishable, which we formally prove in Claim 4.7.

Hybrid H_5^x: In this hybrid, we proceed as in the previous hybrid except that in step (x).2.d the n-out-of-n secret sharing of the input s_{1-b} is changed to an n-out-of-n secret sharing of a random value r (where x is used to denote the 4th or the 6th stage of the simulator, namely $x = 4$ or $x = 6$). The shares $s_{1-b,1}, \dots, s_{1-b,j^*-1}, s_{1-b,j^*+1}, \dots, s_{1-b,m}$ are then used as in the previous hybrid for the generation of the ciphertexts $\mathsf{ct}_c^{0,i}, \mathsf{ct}_c^{1,i}$ for all $i \in [m] \setminus \{j^*\}, c \in \{0,1\}$. The perfect indistinguishability between this and the previous hybrid follows from the security of the n-out-of-n secret sharing scheme, this is formally proven in Claim 4.8

Hybrid H_6: This hybrid corresponds to the simulator described in Figure 5.2.

From the above arguments, we can conclude that $\mathsf{H}_0 \approx \mathsf{H}_1 \approx \mathsf{H}_1' \approx_c \mathsf{H}_2^4 \approx_c \mathsf{H}_3^4 \approx \mathsf{H}_4^4 \approx \mathsf{H}_5^4 \approx_c \mathsf{H}_2^6 \approx_c \mathsf{H}_3^6 \approx \mathsf{H}_4^6 \approx \mathsf{H}_5^6 \approx \mathsf{H}_6$.

□

Figure 4.2: Simulator $\mathsf{Sim}' = (\mathsf{Sim}'^1, \mathsf{Sim}'^2)$ of OT'

Input. The simulator Sim'^1 takes as input the distribution $\mathcal{D}$ of s and 1^λ.

1) First thread: Sim'^1 computes the following steps:
1. Upon receiving $\{\mathsf{ot}_{1,i}, \mathsf{ot}'_{1,i}\}_{i\in[m]}$ from R^* compute $\{\mathsf{ot}_{2,i}, \mathsf{ot}'_{2,i}\}_{i\in[m]}$ and $\mathcal{I}$ as the honest sender would do and send them to R^*.
2. If R^* does not send a third round, abort and output the view of R^*.
3. Upon receiving $\{b_i, r_i\}_{i\in[m]:\mathcal{I}_i=1}$ and $\{d_i, r'_i\}_{i\in[m]:\mathcal{I}_i=0}$ from R^*, add them to the initially empty set $\mathsf{Defense}$, if their are valid w.r.t. the previously received messages $\{\mathsf{ot}_{1,i}, \mathsf{ot}'_{1,i}\}_{i\in[m]}$.

2) Estimate the abort probability: Sim'^1 initializes $\mathsf{ctr} := 0, T := 0$ and computes the following steps:
1. Increment T, i.e. $T := T + 1$.
2. Compute and send a new 2nd round with fresh randomness to R^* as an honest sender would do. If a valid third round is received from R^* increment ctr, i.e. $\mathsf{ctr} := \mathsf{ctr} + 1$.
3. If $\mathsf{ctr} = 12\lambda$ then output $p = \frac{12\lambda}{T}$, otherwise, perform a new rewind, i.e. return to step (2).1.
4. Set $\mathsf{maxrew} = \lceil \frac{\lambda}{p} \rceil$.

3) Extracting the input of R^*: Sim'^1 initializes $\mathsf{ctr} := 0$ and computes the following steps:
1. Perform the rewinds as follows:

Rewinding Threads: Sim'^1 computes the following steps:
(a) Increment ctr, i.e. $\mathsf{ctr} := \mathsf{ctr} + 1$.
(b) Repeat the step (2).2 using fresh randomness, and therefore also a fresh $\mathcal{I}'$, and upon receiving $\{b_i, r_i\}_{i\in[m]:\mathcal{I}'_i=1}$ and $\{d_i, r'_i\}_{i\in[m]:\mathcal{I}'_i=0}$ from R^* check that these values are valid w.r.t. $\{\mathsf{ot}^*_{1,i}, \mathsf{ot}_{1,i}\}_{i\in[m]}$. If this is the case, then add the values that are not already contained in $\mathsf{Defense}$ to $\mathsf{Defense}$.
(c) If $\mathsf{ctr} > \mathsf{maxrew}$ output fail.
(d) Check that there exists at least a single slot j^* such that $\{b_{j^*}, r_{j^*}\}$ and $\{d_{j^*}, r'_{j^*}\}$ are contained in $\mathsf{Defense}$. If this is the case, proceed to next step; else continue rewinding, i.e. return to step (3).1.a.

2. Uses the values $\{b_{j^*}, r_{j^*}\}$ and $\{d_{j^*}, r'_{j^*}\}$ to extract the bit b used by R^* to construct $\mathsf{ot}_{1,j^*}, \mathsf{ot}'_{1,j^*}$, i.e. compute $b := b_{j^*} \oplus d_{j^*}$.

4) Estimate the abort probability: Sim initializes $\mathsf{ctr} := 0, T := 0$ and computes the following steps:
1. Increment T, i.e. $T := T + 1$.
2. Repeat the steps of round 2 using fresh randomness and compute $\{\mathsf{ot}_{2,i}, \mathsf{ot}'_{2,i}\}_{i\in[m]}$ as follows:

(a) Sample s_b, r at random from the distribution $\mathcal{D}$.
(b) Sample $\mathsf{k}^{j^*}_{b_{j^*}} \leftarrow \{0,1\}^\lambda$, $\mathsf{k}^{j^*}_{1\oplus b_{j^*}} := 0^\lambda$ and sample $\mathsf{k}_0^i, \mathsf{k}_1^i \leftarrow \{0,1\}^\lambda$ for all $i \in [m] \setminus \{j^*\}$.
(c) Sample $s_{b,l} \leftarrow \{0,1\}^\lambda$ for all $l \in [m] \setminus \{j^*\}$ and set $s_{b,j^*} := s_b \oplus s_{b,1} \oplus \cdots \oplus s_{b,m}$.
(d) Sample $s_{1-b,l} \leftarrow \{0,1\}^\lambda$ for all $l \in [m] \setminus \{j^*\}$ and set $s_{1-b,j^*} := r \oplus s_{1-b,1} \oplus \cdots \oplus s_{1-b,m}$.
(e) Set $\mathsf{ct}_b^{d_{j^*},j^*} := \mathsf{k}^{j^*}_{d_{j^*}\oplus b_{j^*}} \oplus s_{b_{j^*},j^*}, \mathsf{ct}_{1\oplus b}^{d_{j^*},j^*} \leftarrow \{0,1\}^\lambda$ and $\mathsf{ct}_0^{1\oplus d_{j^*},j^*} := 0^\lambda, \mathsf{ct}_1^{1\oplus d_{j^*},j^*} := 0^\lambda$.
(f) For all $i \in [m] \setminus \{j^*\}$, set $\mathsf{ct}_c^{0,i} := \mathsf{k}_c^i \oplus s_{c,i}$ and $\mathsf{ct}_c^{1,i} := \mathsf{k}_{1\oplus c}^i \oplus s_{c,i}$ for all $i \in [m] \setminus \{j^*\}, c \in \{0,1\}$.

(g) Generate $\mathsf{ot}_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}'_{1,i}, (\mathsf{k}_0^i, \mathsf{k}_1^i))$ and $\mathsf{ot}'_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}_{1,i}, ((\mathsf{ct}_0^{0,i}, \mathsf{ct}_1^{0,i}), (\mathsf{ct}_0^{1,i}, \mathsf{ct}_1^{1,i})))$ for all $i \in [m]$.

If a valid third round is received from R^* increment ctr, i.e. $\mathsf{ctr} := \mathsf{ctr} + 1$.

3. If $\mathsf{ctr} = 12\lambda$ then output $q = \frac{12\lambda}{T}$, otherwise, perform a new rewind, i.e. return to step (4).1.
4. Set $\mathsf{maxrew}' = \lceil \frac{\lambda}{q} \rceil$.

5) Query the ideal functionality: Sim'^1 sets $d := \lambda \cdot \mathsf{maxrew}'$ and queries the ideal functionality using $(b, 1^d)$ to obtain $(s_b^1, \ldots, s_b^d)$.

6) Forcing the output Sim'^2 on input $(s_b^1, \ldots, s_b^d)$ initializes $\mathsf{ctr} := 0$ and computes the following steps:

Forcing the output:
1. Increment ctr, i.e. $\mathsf{ctr} := \mathsf{ctr} + 1$
2. Compute $\{\mathsf{ot}_{2,i}, \mathsf{ot}'_{2,i}\}_{i \in [m]}$ as described in Step 4 "Estimate the abort probability" but using s_b^{ctr} as its input.
3. If $\mathsf{ctr} > \lambda \cdot \mathsf{maxrew}'$ output fail.
4. If R^* outputs an accepting third round, then output the view of R^* in this thread; else continue the rewinding, i.e. return to step (6).1.

Sim also keeps a count of its overall running time and if it reaches 2^λ steps it outputs fail.

Claim 4.3 Sim' *runs in expected polynomial time in* λ.

Proof. The analysis of the simulator follows similarly to the analysis in [GK96] as recapped in [Lin16]. We analyze the running time in more detail. Let ε be the probability that the adversary R^* outputs a valid third round. Now, we analyze the running time of the simulator step by step.

In the first step ("First thread") the simulator clearly runs in polynomial time. The estimation of the probability for the second step ("Estimate the abort probability") is executed until Sim' does not collect 12λ valid third rounds from R^*. Since the probability that R^* outputs a valid third round is ε, Sim' stops after $12\lambda/\varepsilon$ attempts.

From the analysis in [GK96] it follows that the estimation p is within a constant factor from ε, unless of a negligible probability $2^{-\lambda}$ (therefore the case of running 2^λ steps adds only a polynomial amount of overhead to the simulator).

In the third step ("Extracting the input of R^*") the simulator Sim' stops after maxrew, which is $\frac{\mathrm{poly}(\lambda)}{p}$, iterations. The fifth step ("Query the ideal functionality") happens, as the first step, in polynomial time. Relying on the same arguments as in Step 2 and 3, one can argue that also Step 4 ("Estimate the abort probability") and Step 6 ("Forcing the output") have an expected running time of $\frac{\mathrm{poly}(\lambda)}{\varepsilon}$.

From the above analysis, it follows that if the simulator Sim' does not perform any rewinds it runs in strict polynomial time. Taking into account the rewinds, it follows, as described above, that the Steps 2-6 are executed with an expected running time of $\frac{\mathrm{poly}(\lambda)}{\varepsilon}$. This results in an overall running time of the simulator that is expected polynomial in λ. □

Claim 4.4 Sim *outputs* fail *with negligible probability in* λ.

Proof. There are three possibilites that the simulator Sim' outputs fail:

1. The simulator executes more than 2^λ steps
2. The simulator executes more than maxrew rewinds in step 3 ("Extracting the input of R^*")
3. The simulator executes more than maxrew rewinds in step 6 ("Forcing the output")

The first case is directly bounded by the previous proof, i.e. if the simulator runs in expected polynomial time in λ then it only executes more than 2^λ steps with negligible probability.

To show that the second case is negligible, we need to bound the extraction probability of the simulator Sim'. The simulator is able to extract successfully if it is able to collect defenses for $\mathsf{ot}_{1,j^*}, \mathsf{ot}'_{1,j^*}$ for at least a single index $j^* \in [m]$. Since the simulator executes each rewind with new fresh randomness, i.e. the set $\mathcal{I}$ is always randomly sampled, it follows that the simulator Sim overall (in the worst case) collects maxrew third rounds from R^*. Since $\mathcal{I}$ sampled freshly with every rewind, it follows that the probability that the simulator is not able to extract for at least a single index is $\left(\frac{1}{2^m}\right)^{\mathsf{maxrew}}$, which is the same probability as if the same index $\mathcal{I}$ is always queried. Therefore, Sim successfully extracts except with negligible probability.

To bound the probability of the third case we first prove that the probability that R^* replies with a valid third round to a simulated 2nd round (as described in step (4)) is negligible close to the probability that R^* replies with a valid third round to an honestly computed 2nd round of the protocol.

In order to do so we consider the following hybrids:

Hybrid H_1: We start by considering an hybrid were the messages of the sender are computed honestly but the input bits b_{j^*} and d_{j^*} of the receiver's messages $\mathsf{ot}_{1,j^*}, \mathsf{ot}'_{1,j^*}$ are extracted in exponential time to compute $b = d_{j^*} \oplus b_{j^*}$. The rest of the protocol is executed as the honest sender.

Hybrid H_1: In this hybrid, we proceed as in H_0 but the message ot^0_{2,j^*} is generated as described in step (4).g (i.e. using the simulator $\mathsf{Sim}'^2_{j^*,0}$ w.r.t. input $\mathsf{k}^{j^*}_{b_{j^*}} \leftarrow \{0,1\}^\lambda$). For the remaining indices $i \in [m] \setminus \{j^*\}$ both of the keys remain randomly sampled $\mathsf{k}^i_0, \mathsf{k}^i_1 \leftarrow \{0,1\}^\lambda$. The indistinguishability of the hybrids follows from the list simulatable sender security of the underlying OT protocol OT', the proof follows similar to the one described in Claim 5.5.

Hybrid H_2: In this hybrid, we proceed as in the previous hybrid while only sampling one of the keys $\mathsf{k}^{j^*}_0, \mathsf{k}^{j^*}_1$ that are used for the generation of the OT messages ot_{2,j^*} randomly and set the other to the all-zero string. Formally, $\mathsf{k}^{j^*}_{b_{j^*}} \leftarrow \{0,1\}^\lambda$ and $\mathsf{k}^{j^*}_{1\oplus b_{j^*}} := 0^\lambda$. For the remaining indices $i \in [m] \setminus \{j^*\}$ both of the keys remain randomly sampled $\mathsf{k}^i_0, \mathsf{k}^i_1 \leftarrow \{0,1\}^\lambda$. The indistinguishability of this and the previous hybrid follows from the sender privacy of the underlying OT protocol OT. The proof works analogously as the proof of Claim 4.5.

Hybrid H_3: In this hybrid, we proceed as in the previous hybrid except that the ciphertexts $\mathsf{ct}^{d_{j^*},j^*}_0, \mathsf{ct}^{d_{j^*},j^*}_1$ are generated correctly[13] while the ciphertexts for the other slot $\mathsf{ct}^{d_{j^*}\oplus 1,j^*}_0, \mathsf{ct}^{d_{j^*}\oplus 1,j^*}_1$ are set to the all-zero string, i.e. $\mathsf{ct}^{d_{j^*}\oplus 1,j^*}_0 := 0^\lambda, \mathsf{ct}^{d_{j^*}\oplus 1,j^*}_1 := 0^\lambda$. For the remaining indices $i \in [m] \setminus \{j^*\}$ all of the ciphertexts remain correctly generated, i.e. $\mathsf{ct}^{0,i}_c := \mathsf{k}^i_c \oplus s_{c,i}$ and $\mathsf{ct}^{1,i}_c := \mathsf{k}^i_{1\oplus c} \oplus s_{c,i}$ for $c \in \{0,1\}$. The indistinguishability this and the previous hybrid follows, similar to the previous two hybrids, i.e. from the sender privacy of the underlying OT protocol OT. The proof works analogously as the proof of Claim 4.6.

Hybrid H_4: In this hybrid, we proceed as in the previous hybrid except that the ciphertext $\mathsf{ct}^{d_{j^*},j^*}_{1\oplus b} := \mathsf{k}'^{j^*}_{1\oplus b_{j^*}} \oplus s_{1\oplus b,j^*}$ is changed to a randomly sampled ciphertext $\mathsf{ct}^{d_{j^*},j^*}_{1\oplus b} \leftarrow \{0,1\}^\lambda$. For the remaining indices $i \in [m] \setminus \{j^*\}$ all the ciphertexts remain generated as before. Since, due to the previous hybrid, the key $\mathsf{k}^{j^*}_{1\oplus b_{j^*}}/\mathsf{k}'^{j^*}_{1\oplus b_{j^*}}$ is not part of the execution anymore, we can rely on the perfect security of the one-time pad to show that this and the previous hybrid are perfectly indistinguishable. The proof works analogously to the proof of Claim 4.7.

Hybrid H_5: In this hybrid, we proceed as in the previous hybrid except that in step (x).2.d the n-out-of-n secret sharing of the input s_{1-b} is changed to an n-out-of-n secret sharing of a random value r. The shares $s_{1-b,1}, \ldots, s_{1-b,j^*-1}, s_{1-b,j^*+1}, \ldots, s_{1-b,m}$ are then used as in the previous hybrid for the generation of the ciphertexts $\mathsf{ct}^{0,i}_c, \mathsf{ct}^{1,i}_c$ for all $i \in [m] \setminus \{j^*\}, c \in \{0,1\}$. The perfect indistinguishability between this and the previous hybrid follows from the security of the n-out-of-n secret sharing scheme and the proof works analogously to the proof of Claim 4.8

From the above hybrids, we can conclude that the probability that R^* replies with a valid third round to an honestly computed 2nd round, which we denote with ε, is negligible close to the probability that R^* replies with a valid third round to simulated 2nd round (as described in step (4)). Therefore, the probability that R^* replies with a valid third round after receiving a simulated 2nd round is $\varepsilon - \nu(\lambda)$, where $\nu(\lambda)$ is a negligible function. Since we are considering a non-aborting R^* where ε is non-negligible, we can assume that $\varepsilon > 2\nu(\lambda)$. Following the analysis of [Lin16, Claim 5.8 case 2], we can conclude that the simulator executes more than $\lambda \cdot \mathsf{maxrew}'$ rewinds in step 6 only with negligible probability. $\square$

[13] Here, for the generation of the ciphertext $\mathsf{ct}^{d_{j^*},j^*}_{1\oplus b}$, we still use a random key $\mathsf{k}'^{j^*}_{1\oplus b_{j^*}}$ for the encryption and not the all-zero key $\mathsf{k}^{j^*}_{1\oplus b_{j^*}}$.

Hybrid Transitions. Now, we prove the indistinguishability of the real and the ideal world by arguing the different hybrid transitions.

Claim 4.5 *Let* OT *be a private OT protocol, then the hybrids* H_1' *and* H_2^x *are computationally indistinguishable, with* $x = 4$ *and* $x = 6$ *(to be more precise, x is used to denote the 4th or the 6th stage of the simulation).*

Proof. We prove this claim by contradiction. Assume that there exists an adversary $\mathcal{A}'$, that can distinguish between the two hybrids with non-negligible probability, then we can use $\mathcal{A}'$ to construct an adversary $\mathcal{A}$ that breaks the sender privacy of the OT protocol OT.

Let CH be the challenger of the sender's privacy game of the underlying OT protocol OT.

The adversary $\mathcal{A}$ upon receiving $\{\mathsf{ot}_{1,i}, \mathsf{ot}'_{1,i}\}_{i\in[m]}$ from $\mathcal{A}'$ extracts the input of some index j^* as follows: $\mathcal{A}$ continues the execution and if it obtains a valid third round message she rewinds $\mathcal{A}'$ at the beginning of the 2nd rounds, generating honestly executed 2nd round messages with freshly sampled randomness (in particular the string $\mathcal{I}'$ is taken at random). Therefore it holds that a second valid third round has been generated with respect to a string $\mathcal{I}'$ that is different from $\mathcal{I}$ (the string that has been used to obtain the first accepting transcript) with probability $1 - \frac{1}{2^m}$ (the probability that both of the strings are the same is $\frac{1}{2^m}$, therefore the probability that they both are different is $1 - \frac{1}{2^m}$). In the case that the two strings are different, they are at least different in a single position j^*, i.e. $\mathcal{I}_{j^*} = 1$ and $\mathcal{I}'_{j^*} = 0$. If a valid third round is obtained for both of the strings $\mathcal{I}$ and $\mathcal{I}'$, both of the values b_{j^*} and d_{j^*} are known and $b = b_{j^*} \oplus d_{j^*}$ can be computed. Since the adversary outputs an accepting transcript with non-negligible probability, after two rewinds with some non-negligible probability, the reduction obtains two valid transcripts from the adversary and finish the executions as follow. Therefore, the probability that a valid second transcript is received conditioned that a valid first transcript has been received is $p \cdot (1 - \frac{1}{2^m})$.

If the extraction was not successful $\mathcal{A}$ flips a random coins and terminate otherwise $\mathcal{A}$ rewinds $\mathcal{A}'$ at the beginning of the 2nd rounds and compute the message ot_{2,j^*} as follows:

1. The reduction $\mathcal{A}$ samples $\mathsf{k}_c^i, s_c, s_{c,j} \leftarrow \{0,1\}^\lambda$ for all $i \in [m], j \in [m-1], c \in \{0,1\}$ and sets $s_{c,m} := s_c \oplus s_{c,1} \oplus \cdots \oplus s_{c,m-1}$ for both $c \in \{0,1\}$. Afterwards, it computes $\mathsf{ct}_c^{0,i} := \mathsf{k}_c^i \oplus s_{c,i}$ and $\mathsf{ct}_c^{1,i} := \mathsf{k}_{1\oplus c}^i \oplus s_{c,i}$ for all $i \in [m], c \in \{0,1\}$ and generates $\mathsf{ot}_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}_{1,i}, (\mathsf{k}_0^i, \mathsf{k}_1^i))$ for all $i \in [m] \setminus \{j^*\}$ and $\mathsf{ot}'_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}'_{1,i}, ((\mathsf{ct}_0^{0,i}, \mathsf{ct}_1^{0,i}), (\mathsf{ct}_0^{1,i}, \mathsf{ct}_1^{1,i})))$ for all $i \in [m]$.
2. To obtain the OT message ot_{2,j^*}, the adversary $\mathcal{A}$ submits $((\mathsf{k}_0^{j^*}, \mathsf{k}_1^{j^*}), (\mathsf{k}_0'^{j^*}, \mathsf{k}_1'^{j^*}), \mathsf{ot}_{1,j^*})$ with $\mathsf{k}'^{j^*}_{b_{j^*}} = \mathsf{k}^{j^*}_{b_{j^*}}$ and $\mathsf{k}'^{j^*}_{1\oplus b_{j^*}} = 0^\lambda$ to CH which then replies with ot_{2,j^*}. Finally, $\{\mathsf{ot}_{2,i}, \mathsf{ot}'_{2,i}\}_{i\in[m]}$ and $\mathcal{I}$ with $\mathcal{I} \leftarrow \{0,1\}^m$ is sent to $\mathcal{A}'$.
3. $\mathcal{A}$ outputs the view of $\mathcal{A}'$.

If the adversary $\mathcal{A}'$ is now able to distinguish between the two hybrids with non-negligible probability, then our constructed adversary $\mathcal{A}$ breaks the privacy of the underlying OT protocol OT with non-negligible probability. This results in a contradiction and concludes the proof. □

Claim 4.6 *Let* OT *be a private OT protocol, then the hybrids* H_2^x *and* H_3^x *are computationally indistinguishable, with* $x = 4$ *and* $x = 6$ *(to be more precise, x is used to denote the 4th or the 6th stage of the simulation).*

Proof. We prove this claim by contradiction. Assume that there exists an adversary $\mathcal{A}'$, that can distinguish between the two hybrids with non-negligible probability, then we can use $\mathcal{A}'$ to construct an adversary $\mathcal{A}$ that breaks the sender privacy of the OT protocol OT.

The adversary $\mathcal{A}$ upon receiving $\{\mathsf{ot}_{1,i}, \mathsf{ot}'_{1,i}\}_{i\in[m]}$ from $\mathcal{A}'$ extracts the input of some index j^* as follows: $\mathcal{A}$ continues the execution and if it obtains a valid third round message she rewinds $\mathcal{A}'$ at the beginning of the 2nd rounds, generating honestly executed 2nd round messages with freshly sampled randomness (in particular the string $\mathcal{I}'$ is taken at random). Therefore it holds that a second valid third round has been generated with respect to a string $\mathcal{I}'$ that is different from $\mathcal{I}$ (the string that has been used to obtain the first accepting transcript) with probability $1 - \frac{1}{2^m}$ (the probability that both of the strings are the same is $\frac{1}{2^m}$, therefore the probability that they both are different is $1 - \frac{1}{2^m}$). In the case that the two strings are different, they are at least different in a single position j^*, i.e. $\mathcal{I}_{j^*} = 1$ and $\mathcal{I}'_{j^*} = 0$. If a valid third round is obtained for both of the strings $\mathcal{I}$ and $\mathcal{I}'$, both of the values b_{j^*} and d_{j^*} are known and $b = b_{j^*} \oplus d_{j^*}$ can be computed. Since the adversary outputs an accepting transcript with non-negligible probability, after two rewinds with some non-negligible probability, the reduction obtains two valid transcripts from the adversary and finish the executions as follow. Therefore, the probability that a valid second transcript is received conditioned that a valid first transcript has been received is $p \cdot (1 - \frac{1}{2^m})$.

If the extraction was successful $\mathcal{A}$ rewinds $\mathcal{A}'$ at the beginning of the 2nd rounds and compute the message ot'_{2,j^*} as follows:

1. Upon receiving $\{\mathsf{ot}_{1,i}, \mathsf{ot}'_{1,i}\}_{i\in[m]}$ from $\mathcal{A}'$, the reduction $\mathcal{A}$ samples $\mathsf{k}^{j^*}_{b_{j^*}}, \mathsf{k}^i_c, s_c, s_{c,j} \leftarrow \{0,1\}^\lambda$ for all $i \in [m] \setminus \{j^*\}, j \in [m-1], c \in \{0,1\}$ and sets $s_{c,m} := s_c \oplus s_{c,1} \oplus \cdots \oplus s_{c,m-1}$ for both $c \in \{0,1\}$ as well as $\mathsf{k}^{j^*}_{1\oplus b_{j^*}} := 0^\lambda$. It then generates $\mathsf{ot}_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}_{1,i}, (\mathsf{k}^i_0, \mathsf{k}^i_1))$ for all $i \in [m]$. Afterwards, it computes $\mathsf{ct}^{0,i}_c := \mathsf{k}^i_c \oplus s_{c,i}$ and $\mathsf{ct}^{1,i}_c := \mathsf{k}^i_{1\oplus c} \oplus s_{c,i}$ for all $i \in [m] \setminus \{j^*\}, c \in \{0,1\}$ and generates $\mathsf{ot}'_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}'_{1,i}, ((\mathsf{ct}^{0,i}_0, \mathsf{ct}^{0,i}_1), (\mathsf{ct}^{1,i}_0, \mathsf{ct}^{1,i}_1)))$ for all $i \in [m] \setminus \{j^*\}$. Furthermore, the adversary $\mathcal{A}$ computes $\mathsf{ct}^{d_{j^*},j^*}_b := \mathsf{k}^{j^*}_{b_{j^*}} \oplus s_{b,j^*}$ and $\mathsf{ct}^{d_{j^*},j^*}_{1\oplus b} := \mathsf{k}'^{j^*}_{1\oplus b_{j^*}} \oplus s_{1\oplus b,j^*}, \mathsf{ct}^{1\oplus d_{j^*},j^*}_0 := \mathsf{k}'^{j^*}_{1\oplus b_{j^*}} \oplus s_{0,j^*}, \mathsf{ct}^{1\oplus d_{j^*},j^*}_1 := \mathsf{k}^{j^*}_{b_{j^*}} \oplus s_{1,j^*}$, where $\mathsf{k}'^{j^*}_{1\oplus b_{j^*}} \leftarrow \{0,1\}^\lambda$. Afterwards, $\mathcal{A}$ sets $\mathsf{ct}'^{d_{j^*},j^*}_c := \mathsf{ct}^{d_{j^*},j^*}_c$ and $\mathsf{ct}'^{1\oplus d_{j^*},j^*}_c := 0^\lambda$ for $c \in \{0,1\}$.
2. To obtain the OT message ot'_{2,j^*}, the adversary $\mathcal{A}$ submits $(((\mathsf{ct}^{0,j^*}_0, \mathsf{ct}^{0,j^*}_1), (\mathsf{ct}^{1,j^*}_0, \mathsf{ct}^{1,j^*}_1)), ((\mathsf{ct}'^{0,j^*}_0, \mathsf{ct}'^{0,j^*}_1), (\mathsf{ct}'^{1,j^*}_0, \mathsf{ct}'^{1,j^*}_1)), \mathsf{ot}'_{1,j^*})$ to CH which then replies with ot'_{2,j^*}. Finally, $\{\mathsf{ot}_{2,i}, \mathsf{ot}'_{2,i}\}_{i\in[m]}$ and $\mathcal{I}$ with $\mathcal{I} \leftarrow \{0,1\}^m$ are being sent to $\mathcal{A}'$.
3. $\mathcal{A}$ outputs the view of $\mathcal{A}'$.

If the adversary $\mathcal{A}'$ is now able to distinguish between the two hybrids with non-negligible probability, then our constructed adversary $\mathcal{A}$ breaks the sender privacy of the underlying OT protocol OT with non-negligible probability. This results in a contradiction and concludes the proof. □

Claim 4.7 *The hybrids* H^x_3 *and* H^x_4 *are perfectly indistinguishable due to the security of one-time pad, for* $x = 4$ *and* $x = 6$ *(to be more precise, x is used to denote the 4th or the 6th stage of the simulation).*

Proof. We prove this claim by contradiction. Assume that there exists an adversary $\mathcal{A}'$, that can distinguish between the two hybrids with non-negligible probability, then we can use $\mathcal{A}'$ to construct an adversary $\mathcal{A}$ that breaks the one-time pad. Let CH be the challenger of the one-time pad.

1. Upon receiving $\{\mathsf{ot}_{1,i}, \mathsf{ot}'_{1,i}\}_{i\in[m]}$ from $\mathcal{A}'$, $\mathcal{A}$ apply Ext (which exists due to the security of OT) and extracts the input of the receiver some random position j^*. Let b_{j^*} the input extracted from ot_{1,j^*} and d_{j^*} the input extracted from ot'_{1,j^*}.
   Then the reduction $\mathcal{A}$ samples $\mathsf{k}^{j^*}_{b_{j^*}}, \mathsf{k}^i_c, s_c, s_{c,j} \leftarrow \{0,1\}^\lambda$ for all $i \in [m] \setminus \{j^*\}, j \in [m-1], c \in \{0,1\}$ and sets $s_{c,m} := s_c \oplus s_{c,1} \oplus \cdots \oplus s_{c,m-1}$ for both $c \in \{0,1\}$ as well as $\mathsf{k}^{j^*}_{1\oplus b_{j^*}} := 0^\lambda$. It then generates $\mathsf{ot}_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}_{1,i}, (\mathsf{k}^i_0, \mathsf{k}^i_1))$ for all $i \in [m]$. Afterwards, it computes $\mathsf{ct}^{0,i}_c := \mathsf{k}^i_c \oplus s_{c,i}$ and $\mathsf{ct}^{1,i}_c := \mathsf{k}^i_{1\oplus c} \oplus s_{c,i}$ for all $i \in [m] \setminus \{j^*\}, c \in \{0,1\}$ and generates $\mathsf{ot}'_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}'_{1,i}, ((\mathsf{ct}^{0,i}_0, \mathsf{ct}^{0,i}_1), (\mathsf{ct}^{1,i}_0, \mathsf{ct}^{1,i}_1)))$ for all $i \in [m] \setminus \{j^*\}$. Furthermore, the adversary $\mathcal{A}$ computes $\mathsf{ct}'^{d_{j^*},j^*}_b := \mathsf{k}^{j^*}_{b_{j^*}} \oplus s_{b,j^*}$ and sets $\mathsf{ct}'^{1\oplus d_{j^*},j^*}_c := 0^\lambda$ for $c \in \{0,1\}$.
2. To obtain the ciphertext $\mathsf{ct}'^{d_{j^*},j^*}_{1\oplus b}$, the adversary $\mathcal{A}$ submits (s_{1-b,j^*}, r) with $r \leftarrow \{0,1\}^\lambda$ to the underlying challenger CH of the one-time pad. Afterwards, the adversary $\mathcal{A}$ computes $\mathsf{ot}'_{2,j^*} \leftarrow \mathsf{OT}_2(\mathsf{ot}'_{1,j^*}, ((\mathsf{ct}'^{0,j^*}_0, \mathsf{ct}'^{0,j^*}_1), (\mathsf{ct}'^{1,j^*}_0, \mathsf{ct}'^{1,j^*}_1)))$ and sends $\{\mathsf{ot}_{2,i}, \mathsf{ot}'_{2,i}\}_{i\in[m]}$ and $\mathcal{I}$ with $\mathcal{I} \leftarrow \{0,1\}^m$ to $\mathcal{A}'$.
3. $\mathcal{A}$ outputs the view of $\mathcal{A}'$.

If the adversary $\mathcal{A}'$ is now able to distinguish between the two hybrids with non-negligible probability, then our constructed adversary $\mathcal{A}$ breaks the privacy of the underlying one-time pad with non-negligible probability. This results in a contradiction and concludes the proof. □

Claim 4.8 *The hybrids H_4^x and H_5^x are perfectly indistinguishable due to the security of n-out-of-n secret sharing scheme, for $x = 4$ and $x = 6$ (to be more precise, x is used to denote the 4th or the 6th stage of the simulation).*

Proof. We prove this claim by contradiction. Assume that there exists an adversary $\mathcal{A}'$, that can distinguish between the two hybrids H_4 and H_5 with non-negligible probability, then we can use $\mathcal{A}'$ to construct an adversary $\mathcal{A}$ that breaks the n-out-of-n secret sharing scheme. Let CH be the challenger of the n-out-of-n secret sharing scheme.

The adversary $\mathcal{A}$ behaves as follows:

1. To obtain the shares $\{s_{1-b,i}\}_{i\in[m]\setminus\{j^*\}}$ for the input s_{1-b}, the adversary $\mathcal{A}$ submits (s_{1-b}, r) with $r \leftarrow \{0,1\}^\lambda$ to the underlying challenger CH of the n-out-of-n secret sharing scheme.
2. Upon receiving $\{\mathsf{ot}_{1,i}, \mathsf{ot}'_{1,i}\}_{i\in[m]}$ from $\mathcal{A}'$, $\mathcal{A}$ apply Ext (which exists due to the security of OT) and extracts the input of the receiver in some random position j^*. Let b_{j^*} the input extracted from ot_{1,j^*} and d_{j^*} the input extracted from ot'_{1,j^*}.
   Then the reduction $\mathcal{A}$ samples $\mathsf{k}^{j^*}_{b_{j^*}}, \mathsf{k}^i_c, s_b, s_{b,j} \leftarrow \{0,1\}^\lambda$ for all $i \in [m] \setminus \{j^*\}, j \in [m-1], c \in \{0,1\}$ and sets $s_{b,m} := s_b \oplus s_{b,1} \oplus \cdots \oplus s_{b,m-1}$ as well as $\mathsf{k}^{j^*}_{1\oplus b_{j^*}} := 0^\lambda$. It then generates $\mathsf{ot}_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}_{1,i}, (\mathsf{k}^i_0, \mathsf{k}^i_1))$ for all $i \in [m]$. Afterwards, the adversary $\mathcal{A}$ computes $\mathsf{ct}^{0,i}_c := \mathsf{k}^i_c \oplus s_{c,i}$ and $\mathsf{ct}^{1,i}_c := \mathsf{k}^i_{1\oplus c} \oplus s_{c,i}$ for all $i \in [m] \setminus \{j^*\}, c \in \{0,1\}$ as well as $\mathsf{ct}'^{d_{j^*},j^*}_b := \mathsf{k}^{j^*}_{b_{j^*}} \oplus s_{b,j^*}$ and $\mathsf{ct}'^{d_{j^*},j^*}_{1\oplus b} \leftarrow \{0,1\}^\lambda$ and sets $\mathsf{ct}'^{1\oplus d_{j^*},j^*}_c := 0^\lambda$ for $c \in \{0,1\}$. I then generates $\mathsf{ot}'_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}'_{1,i}, ((\mathsf{ct}^{0,i}_0, \mathsf{ct}^{0,i}_1), (\mathsf{ct}^{1,i}_0, \mathsf{ct}^{1,i}_1)))$ for all $i \in [m]$. In the last step, $\mathcal{A}$ sends $\{\mathsf{ot}_{2,i}, \mathsf{ot}'_{2,i}\}_{i\in[m]}$ and $\mathcal{I}$ with $\mathcal{I} \leftarrow \{0,1\}^m$ to $\mathcal{A}'$.
3. $\mathcal{A}$ outputs the same values that have been output by $\mathcal{A}'$.

If the adversary $\mathcal{A}'$ is now able to distinguish between the two hybrids with non-negligible probability, then our constructed adversary $\mathcal{A}$ breaks the n-out-of-n secret sharing scheme with non-negligible probability. This results in a contradiction and concludes the proof. □

4.2 Indistinguishability Security against Sometimes aborting R^*.

In this section, we state and prove Lemma 4.9, the indistinguishability against an aborting R^*.

Lemma 4.9. *Let OT be a two-round maliciously private OT protocol, then the OT protocol OT' described in Figure 4.1 is a three-round sender private OT protocol against an aborting R^* (accordingly to Definition 2.6).*

Proof. We prove the sender indistinguishability by relying on a sequence of hybrids:

H_0**:** This hybrid corresponds to the execution of OT' using the strings (s_0, s_1) as the input of the sender.

H'_0**:** This hybrid corresponds to the previous one but for a random slot j^* the input bits of the receiver for that slot are extracted using Ext of OT. In particular upon receiving $\{\mathsf{ot}_{1,i}, \mathsf{ot}'_{1,i}\}_{i\in[m]}$ from R^* the hybrid executes Ext on ot_{1,j^*} to extract b_{j^*} and then executed on ot_{1,j^*} to extract d_{j^*}. This hybrid is distributed as the previous one due to the security of OT.

H_1**:** This hybrid corresponds to the previous one but in slot j^* the input $(\mathsf{k}^{j^*}_{b_i}, \mathsf{k}^{j^*}_{b_i})$ instead of $(\mathsf{k}^{j^*}_0, \mathsf{k}^{j^*}_0)$ is used by the sender for the generation of ot_{2,j^*}. The indistinguishability between this and the previous hybrid follows from the sender privacy of OT and is formally shown in Claim 4.10.

H_2**:** This hybrid behaves as the previous hybrid, with the difference that besides using $(\mathsf{k}^{j^*}_{b_i}, \mathsf{k}^{j^*}_{b_i})$ for the generation of ot_{2,j^*}, also the input for the generation of ot'_{2,j^*} is changed from $((\mathsf{ct}^{0,j^*}_0, \mathsf{ct}^{0,j^*}_1), (\mathsf{ct}^{1,j^*}_0, \mathsf{ct}^{1,j^*}_1))$ to $((\mathsf{ct}^{d_{j^*},j^*}_0, \mathsf{ct}^{d_{j^*},j^*}_1), (\mathsf{ct}^{d_{j^*},j^*}_0, \mathsf{ct}^{d_{j^*},j^*}_1))$. The indistinguishability between this and the previous hybrid follows from the sender privacy of OT and is formally shown in Claim 4.11.

H_3**:** This hybrid behaves as the previous hybrid, with the difference that the ciphertext $\mathsf{ct}_{1\oplus b}^{d_{j^*},j^*}$ is generated using s'_{b,j^*} instead of $s_{1\oplus b,j^*}$, where s'_{b,j^*} is part of a secret sharing for s_b instead of $s_{1\oplus b}$. The indistinguishability between this hybrid and the previous hybrid follows from the security of the one-time pad and is formally shown in Claim 4.12.

H_4**:** This hybrid corresponds to the execution of OT' using the strings (s_b, s_b) as the input of the sender. It behaves as the previous hybrids, with the difference that the n-out-of-n secret sharing of the input s_{1-b} is changed to an n-out-of-n secret sharing of the input s_b. The shares $s_{1-b,1}, \dots, s_{1-b,j^*-1}, s_{1-b,j^*+1}, \dots, s_{1-b,m}$ are then used as in the previous hybrid for the generation of the ciphertexts $\mathsf{ct}_c^{0,i}, \mathsf{ct}_c^{1,i}$ for all $i \in [m] \setminus \{j^*\}, c \in \{0,1\}$. The perfect indistinguishability between this and the previous hybrid follows from the security of the n-out-of-n secret sharing scheme and is formally proven in Claim 4.12.

The extractor of OT' is defined as the extractor Ext of OT executed on ot_{1,j^*} to extract b_{j^*} and then executed on ot_{1,j^*} to extract d_{j^*}. The final output b is then computed by calculating $b := b_{i^*} \oplus d_{i^*}$.

From the above arguments we can conclude that $\mathsf{H}_0 \approx\approx \mathsf{H}'_0 \approx_c \mathsf{H}_1 \approx_c \mathsf{H}_2 \approx \mathsf{H}_3$. □

Claim 4.10 *Let* OT *be a sender indistinguishable OT protocol, then the hybrids* H_1 *and* H'_0 *are indistinguishable.*

Proof. This proof is very similar to the proof of Claim 4.6.

We prove this claim by contradiction. Assume that there exists an adversary $\mathcal{A}'$, that can distinguish between the two hybrids with non-negligible probability. Let $\{\mathsf{ot}_{1,i}, \mathsf{ot}'_{1,i}\}_{i\in[m]}$ the first round that maximizes the probability of distinguish of $\mathcal{A}'$. Due to the security of OT is possible to apply extractor Ext to extract the input of the receiver from $\{\mathsf{ot}_{1,i}, \mathsf{ot}'_{1,i}\}_{i\in[m]}$. Let b_{j^*} the input extracted from ot_{1,j^*} and d_{j^*} the input extracted from ot'_{1,j^*}.

We can use $\mathcal{A}'$ to construct an adversary $\mathcal{A}$ that breaks the sender privacy of the OT protocol OT, the polynomial time $\mathcal{A}$ runs using auxiliary input $(\{\mathsf{ot}_{1,i}, \mathsf{ot}'_{1,i}\}_{i\in[m]}, b_{j^*}, d_{j^*})$.

Let CH be the challenger of the sender's privacy game of the underlying OT protocol OT.

For the generation of the message ot_{2,j^*}, the adversary $\mathcal{A}$ behaves as follows:

1. The reduction $\mathcal{A}$ activates $\mathcal{A}'$ from round 2 and samples $\mathsf{k}_c^i, s_c, s_{c,j} \leftarrow \{0,1\}^\lambda$ for all $i \in [m], j \in [m-1], c \in \{0,1\}$ and sets $s_{c,m} := s_c \oplus s_{c,1} \oplus \cdots \oplus s_{c,m-1}$ for both $c \in \{0,1\}$. Afterwards, it computes $\mathsf{ct}_c^{0,i} := \mathsf{k}_c^i \oplus s_{c,i}$ and $\mathsf{ct}_c^{1,i} := \mathsf{k}_{1\oplus c}^i \oplus s_{c,i}$ for all $i \in [m], c \in \{0,1\}$ and generates $\mathsf{ot}_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}_{1,i}, (\mathsf{k}_0^i, \mathsf{k}_1^i))$ for all $i \in [m] \setminus \{j^*\}$ and $\mathsf{ot}'_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}'_{1,i}, ((\mathsf{ct}_0^{0,i}, \mathsf{ct}_1^{0,i}), (\mathsf{ct}_0^{1,i}, \mathsf{ct}_1^{1,i})))$ for all $i \in [m]$.
2. To obtain the OT message ot_{2,j^*}, the adversary $\mathcal{A}$ submits $((\mathsf{k}_0^{j^*}, \mathsf{k}_1^{j^*}), \mathsf{ot}_{1,j^*})$ to CH which then replies with ot_{2,j^*}. Finally, $\{\mathsf{ot}_{2,i}, \mathsf{ot}'_{2,i}\}_{i\in[m]}$ and $\mathcal{I}$ with $\mathcal{I} \leftarrow \{0,1\}^m$ is sent to $\mathcal{A}'$.

If the adversary $\mathcal{A}'$ is now able to distinguish between the two hybrids with non-negligible probability, then our constructed adversary $\mathcal{A}$ breaks the privacy of the underlying OT protocol OT with non-negligible probability. This results in a contradiction and concludes the proof. □

Claim 4.11 *Let* OT *be a sender indistinguishable OT protocol, then the hybrids* H_2 *and* H_3 *are indistinguishable.*

Proof. This proof is very similar to the proof of Claim 4.6.

We prove this claim by contradiction. Assume that there exists an adversary $\mathcal{A}'$, that can distinguish between the two hybrids H_1 and H_2 with non-negligible probability. Let $\{\mathsf{ot}_{1,i}, \mathsf{ot}'_{1,i}\}_{i\in[m]}$ the first round that maximizes the probability of distinguish of $\mathcal{A}'$. Due to the security of OT is possible to apply extractor Ext to extract the input of the receiver from $\{\mathsf{ot}_{1,i}, \mathsf{ot}'_{1,i}\}_{i\in[m]}$. Let b_{j^*} the input extracted from ot_{1,j^*} and d_{j^*} the input extracted from ot'_{1,j^*}.

We can use $\mathcal{A}'$ to construct an adversary $\mathcal{A}$ that breaks the sender privacy of the OT protocol OT, the polynomial time $\mathcal{A}$ runs using auxiliary input $(\{\mathsf{ot}_{1,i}, \mathsf{ot}'_{1,i}\}_{i\in[m]}, b_{j^*}, d_{j^*})$. Let CH be the challenger of the sender's privacy game of the underlying OT protocol OT.

For the generation of the message ot'_{2,j^*}, the adversary $\mathcal{A}$ behaves as follows:

1. The reduction $\mathcal{A}$ activates $\mathcal{A}'$ from round 2 and samples $\mathsf{k}_{b_{j^*}}^{j^*}, \mathsf{k}_c^i, s_c, s_{c,j} \leftarrow \{0,1\}^\lambda$ for all $i \in [m] \setminus \{j^*\}, j \in [m-1], c \in \{0,1\}$ and sets $s_{c,m} := s_c \oplus s_{c,1} \oplus \cdots \oplus s_{c,m-1}$ for both $c \in \{0,1\}$ as well as $\mathsf{k}_{1\oplus b_{j^*}}^{j^*} := \mathsf{k}_{b_{j^*}}^{j^*}$. It then generates $\mathsf{ot}_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}_{1,i}, (\mathsf{k}_0^i, \mathsf{k}_1^i))$ for all $i \in [m]$. Afterwards, it computes $\mathsf{ct}_c^{0,i} := \mathsf{k}_c^i \oplus s_{c,i}$ and $\mathsf{ct}_c^{1,i} := \mathsf{k}_{1\oplus c}^i \oplus s_{c,i}$ for all $i \in [m] \setminus \{j^*\}, c \in \{0,1\}$ and generates $\mathsf{ot}'_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}'_{1,i}, ((\mathsf{ct}_0^{0,i}, \mathsf{ct}_1^{0,i}), (\mathsf{ct}_0^{1,i}, \mathsf{ct}_1^{1,i})))$ for all $i \in [m] \setminus \{j^*\}$. Furthermore, the adversary $\mathcal{A}$ computes $\mathsf{ct}_b^{d_{j^*},j^*} := \mathsf{k}_{b_{j^*}}^{j^*} \oplus s_{b,j^*}$ and $\mathsf{ct}_{1\oplus b}^{d_{j^*},j^*} := \mathsf{k}'^{j^*}_{1\oplus b_{j^*}} \oplus s_{1\oplus b,j^*}, \mathsf{ct}_0^{1\oplus d_{j^*},j^*} := \mathsf{k}'^{j^*}_{1\oplus b_{j^*}} \oplus s_{0,j^*}, \mathsf{ct}_1^{1\oplus d_{j^*},j^*} := \mathsf{k}_{b_{j^*}}^{j^*} \oplus s_{1,j^*}$, where $\mathsf{k}'^{j^*}_{1\oplus b_{j^*}} \leftarrow \{0,1\}^\lambda$.
2. To obtain the OT message ot'_{2,j^*}, the adversary $\mathcal{A}$ submits $(((\mathsf{ct}_0^{0,j^*}, \mathsf{ct}_1^{0,j^*}), (\mathsf{ct}_0^{1,j^*}, \mathsf{ct}_1^{1,j^*})), \mathsf{ot}'_{1,j^*})$ to CH which then replies with ot'_{2,j^*}. Finally, $\{\mathsf{ot}_{2,i}, \mathsf{ot}'_{2,i}\}_{i\in[m]}$ and $\mathcal{I}$ with $\mathcal{I} \leftarrow \{0,1\}^m$ are being sent to $\mathcal{A}'$.

If the adversary $\mathcal{A}'$ is now able to distinguish between the two hybrids with non-negligible probability, then our constructed adversary $\mathcal{A}$ breaks the sender privacy of the underlying OT protocol OT with non-negligible probability. This results in a contradiction and concludes the proof. □

Claim 4.12 *The hybrids* H_2 *and* H_3 *are indistinguishable due to the security of one-time pad.*

Proof. This proof is very similar to the proof of Claim 4.7.

We prove this claim by contradiction. Assume that there exists an adversary $\mathcal{A}'$, that can distinguish between the two hybrids H_2 and H_3 with non-negligible probability, then we can use $\mathcal{A}'$ to construct an (unbounded) adversary $\mathcal{A}$ that breaks the one-time pad. Let CH be the challenger of the one-time pad.

The adversary $\mathcal{A}$ behaves as follows:

1. Upon receiving $\{\mathsf{ot}_{1,i}, \mathsf{ot}'_{1,i}\}_{i\in[m]}$ from $\mathcal{A}'$, extracts the input b_{j^*} of ot_{1,j^*} and the input d_{j^*} of ot'_{1,j^*} using Ext and sets $b = b_{j^*} \oplus d_{j^*}$. Then the reduction $\mathcal{A}$ samples $\mathsf{k}_{b_{j^*}}^{j^*}, \mathsf{k}_c^i, s_c, s_{c,j} \leftarrow \{0,1\}^\lambda$ for all $i \in [m] \setminus \{j^*\}, j \in [m-1], c \in \{0,1\}$ and sets $s_{c,m} := s_c \oplus s_{c,1} \oplus \cdots \oplus s_{c,m-1}$ for both $c \in \{0,1\}$ as well as $\mathsf{k}_{1\oplus b_{j^*}}^{j^*} := \mathsf{k}_{b_{j^*}}^{j^*}$. Furthermore, it samples s'_{b,j^*} such that $s'_{b,j^*} := s_{1\oplus b} \bigoplus_{i\in[m]\setminus\{j^*\}} s_{1\oplus b,i}$. It then generates $\mathsf{ot}_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}_{1,i}, (\mathsf{k}_0^i, \mathsf{k}_1^i))$ for all $i \in [m]$. Afterwards, it computes $\mathsf{ct}_c^{0,i} := \mathsf{k}_c^i \oplus s_{c,i}$ and $\mathsf{ct}_c^{1,i} := \mathsf{k}_{1\oplus c}^i \oplus s_{c,i}$ for all $i \in [m] \setminus \{j^*\}, c \in \{0,1\}$ and generates $\mathsf{ot}'_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}'_{1,i}, ((\mathsf{ct}_0^{0,i}, \mathsf{ct}_1^{0,i}), (\mathsf{ct}_0^{1,i}, \mathsf{ct}_1^{1,i})))$ for all $i \in [m] \setminus \{j^*\}$. Furthermore, the adversary $\mathcal{A}$ computes $\mathsf{ct}'^{d_{j^*},j^*}_b := \mathsf{k}_{b_{j^*}}^{j^*} \oplus s'_{b,j^*}$.
2. To obtain the ciphertext $\mathsf{ct}_{1\oplus b}^{d_{j^*},j^*}$, the adversary $\mathcal{A}$ submits (s_{1-b,j^*}, r) to the underlying challenger CH of the one-time pad. Afterwards, the adversary $\mathcal{A}$ computes $\mathsf{ot}'_{2,j^*} \leftarrow \mathsf{OT}_2(\mathsf{ot}'_{1,j^*}, ((\mathsf{ct}_0^{d_{j^*},j^*}, \mathsf{ct}_1^{d_{j^*},j^*}), (\mathsf{ct}_0^{d_{j^*},j^*}, \mathsf{ct}_1^{d_{j^*},j^*})))$ and sends $\{\mathsf{ot}_{2,i}, \mathsf{ot}'_{2,i}\}_{i\in[m]}$ and $\mathcal{I}$ with $\mathcal{I} \leftarrow \{0,1\}^m$ to $\mathcal{A}'$.

If the adversary $\mathcal{A}'$ is now able to distinguish between the two hybrids with non-negligible probability, then our constructed adversary $\mathcal{A}$ breaks the privacy of the underlying one-time pad with non-negligible probability. This results in a contradiction and concludes the proof. □

Claim 4.13 *The hybrids* H_3 *and* H_4 *are indistinguishable due to the security of the* n*-out-of-*n *secret sharing scheme.*

Proof. This proof is very similar to the proof of Claim 4.8.

We prove this claim by contradiction. Assume that there exists an adversary $\mathcal{A}'$, that can distinguish between the two hybrids H_2 and H_3 with non-negligible probability, then we can use $\mathcal{A}'$ to construct an (unbounded) adversary $\mathcal{A}$ that breaks the security of the n-out-of-n secret sharing scheme. Let CH be the challenger of the one-time pad.

The adversary $\mathcal{A}$ behaves as follows:

1. Upon receiving $\{\mathsf{ot}_{1,i}, \mathsf{ot}'_{1,i}\}_{i\in[m]}$ from $\mathcal{A}'$, extracts the input b_{j^*} of ot_{1,j^*} and the input d_{j^*} of ot'_{1,j^*} using Ext and sets $b = b_{j^*} \oplus d_{j^*}$. Then the reduction $\mathcal{A}$ samples $\mathsf{k}_{b_{j^*}}^{j^*}, \mathsf{k}_c^i, s_c, s_{b,j} \leftarrow \{0,1\}^\lambda$ for all $i \in [m] \setminus \{j^*\}, j \in [m-1], c \in \{0,1\}$ and sets $s_{b,m} := s_b \oplus s_{b,1} \oplus \cdots \oplus s_{b,m-1}$ as well as $\mathsf{k}_{1\oplus b_{j^*}}^{j^*} := \mathsf{k}_{b_{j^*}}^{j^*}$ and $\mathsf{ct}_{1\oplus b}^{d_{j^*},j^*} \leftarrow \{0,1\}^\lambda$.

2. To obtain the shares $s_{1-b,1}, \dots, s_{1-b,j^*-1}, s_{1-b,j^*+1}, \dots, s_{1-b,m}$, the adversary $\mathcal{A}$ submits (s_{1-b}, s_b) to the underlying challenger CH of the one-time pad.
3. Then the adversary $\mathcal{A}$ generates $\mathsf{ot}_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}_{1,i}, (\mathsf{k}_0^i, \mathsf{k}_1^i))$ for all $i \in [m]$. Afterwards, it computes $\mathsf{ct}_c^{0,i} := \mathsf{k}_c^i \oplus s_{c,i}$ and $\mathsf{ct}_c^{1,i} := \mathsf{k}_{1\oplus c}^i \oplus s_{c,i}$ for all $i \in [m] \setminus \{j^*\}, c \in \{0,1\}$ and generates $\mathsf{ot}'_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}'_{1,i}, ((\mathsf{ct}_0^{0,i}, \mathsf{ct}_1^{0,i}), (\mathsf{ct}_0^{1,i}, \mathsf{ct}_1^{1,i})))$ for all $i \in [m] \setminus \{j^*\}$. Finally, the adversary $\mathcal{A}$ computes $\mathsf{ct}_b'^{d_{j^*},j^*} := \mathsf{k}_{b_{j^*}}^{j^*} \oplus s'_{b,j^*}$ and $\mathsf{ot}'_{2,j^*} \leftarrow \mathsf{OT}_2(\mathsf{ot}'_{1,j^*}, ((\mathsf{ct}_0^{d_{j^*},j^*}, \mathsf{ct}_1^{d_{j^*},j^*}), (\mathsf{ct}_0^{d_{j^*},j^*}, \mathsf{ct}_1^{d_{j^*},j^*})))$ and sends $\{\mathsf{ot}_{2,i}, \mathsf{ot}'_{2,i}\}_{i\in[m]}$ and $\mathcal{I}$ with $\mathcal{I} \leftarrow \{0,1\}^m$ to $\mathcal{A}'$.

If the adversary $\mathcal{A}'$ is now able to distinguish between the two hybrids with non-negligible probability, then our constructed adversary $\mathcal{A}$ breaks the security of the underlying n-out-of-n with non-negligible probability. This results in a contradiction and concludes the proof. □

4.3 Receiver Privacy Against a Malicious Sender

In this section, we prove Lemma 4.14, i.e. the receiver privacy against a malicious sender.

Lemma 4.14. *Let* OT *be a two-round maliciously private OT protocol, then the OT protocol* OT' *described in Figure 4.1 is a three-round receiver private OT protocol (accordingly to Definition 2.6).*

Proof. We prove the receiver indistinguishability by relying on a sequence of hybrids:

H_0**:** This hybrid corresponds to the execution of OT' using the bit 0 as the input of the receiver.
H_k**:** This hybrid corresponds to the execution of OT' where in the first k executions of OT the bits $d_i = 1 \oplus b_i$ instead of $d_i = b_i$ are used.
H_m**:** This hybrid corresponds to the execution of OT' using the bit 1 as the input of the receiver.

In Claim 5.9, we show that $\mathsf{H}_{k-1} \approx \mathsf{H}_k$ for all $k \in [m]$, which concludes the proof of the lemma. □

Claim 4.15 *Let* OT *be a receiver private OT protocol, then the hybrids* H_{k-1} *and* H_k *are computationally indistinguishable for any* $k \in [m]$.

Proof. For simplicity, we show that H_0 and H_1 are computationally indistinguishable. Our argument carries over straightforwardly to the general case.

We prove this claim by contradiction. We assume that there exists an adversary $\mathcal{A}'$, that can distinguish between the two hybrids H_0 and H_1 with non-negligible probability, then we can use $\mathcal{A}'$ to construct an adversary $\mathcal{A}$ that breaks the receiver privacy of the OT protocol OT.

To construct $\mathcal{A}$, we first guess the index $\mathcal{I}_1$ that $\mathcal{A}'$ outputs in the second round of the protocol execution by randomly flipping a coin. This guess will be correct with probability 1/2, since there exist 2^{m-1} strings of length m where the first bit is 0 and 2^{m-1} strings of length m where the first bit is 0. In the case that $\mathcal{A}$ does not guess $\mathcal{I}_1$ correct, it aborts sampling random coins. Otherwise it proceeds as follows. It samples a random $b_1 \leftarrow \{0,1\}$ and sets $d_1 = b_1$. In the case that $\mathcal{I}_1 = 0$, it sends $(b_1, 1 \oplus b_1)$ to its challenger to obtain $\mathsf{ot}_{1,1}$. In the case that $\mathcal{I}_1 = 1$, it sends $(d_1, 1 \oplus d_1)$ to its challenger to obtain $\mathsf{ot}'_{1,1}$. For the generation of the remaining messages, $\mathcal{A}$ behaves as in the protocol description.

We conclude the reduction with the observation that if the challenger of the OT protocol OT generates the OT message $\mathsf{ot}_{1,1}$ using b_1, in the case that $\mathcal{I}_1 = 0$, or the OT message $\mathsf{ot}'_{1,1}$ using d_1, in the case that $\mathcal{I}_1 = 1$, then hybrid H_0 is simulated and if the challenger generates $\mathsf{ot}_{1,1}$ using $1 \oplus b_1$, in the case that $\mathcal{I}_1 = 0$, or the OT message $\mathsf{ot}'_{1,1}$ using $1 \oplus d_1$, in the case that $\mathcal{I}_1 = 1$, then hybrid H_1 is simulated. If the adversary $\mathcal{A}'$ is now able to distinguish between H_0 and H_1 with non-negligible probability, then our constructed adversary $\mathcal{A}$ breaks the receiver privacy of the underlying OT protocol OT with non-negligible probability.

This results in a contradiction and concludes the proof. □

5 Sender List Simulatable and Rewind-Secure Receiver Private OT

In this section, we present a transformation that turns a three-round oblivious transfer (OT) protocol that is receiver private and sender list simulatable (against sometimes aborting adversaries) in the alternating message model into a three-round OT protocol that is still sender side list simulatable in the same model but moreover is B-rewind receiver-private.

The transformation makes use of the oblivious transfer protocol $\mathsf{OT}' = (\mathsf{OT}'_1, \mathsf{OT}'_2, \mathsf{OT}'_3, (\mathsf{OT}'^S_4, \mathsf{OT}'^R_4))$ described in Figure 4.1 (where we consider a single execution of the scheme without repetition).

Figure 5.1: 3-round sender list-simulatable OT and B-rewind receiver private $\mathsf{OT}^{\mathsf{ListRew}}$

Initialization: The sender's input is s_0, s_1. The receiver uses as its input a bit b. The parties also receive a common parameter $m := \text{poly}(\lambda)$.

Round 1 (Receiver).
1. Sample $b_i \leftarrow \{0,1\}$ and compute $d_i = b_i \oplus b$ for all $i \in [m]$.
2. For all $i \in [m]$ generate $\mathsf{ot}^0_{1,i} := \mathsf{OT}'_1(1^\lambda, b_i)$ and $\mathsf{ot}^1_{1,i}. := \mathsf{OT}'_1(1^\lambda, d_i)$;
3. Send $\{\mathsf{ot}^0_{1,i}, \mathsf{ot}^1_{1,i}\}_{i\in[m]}$ to S.

Round 2 (Sender).
1. Sample $\mathsf{k}^i_c, s_c, s_{c,j} \leftarrow \{0,1\}^\lambda$ for all $i \in [m], j \in [m-1], c \in \{0,1\}$. Set $s_{c,m} := s_c \oplus s_{c,1} \oplus \cdots \oplus s_{c,m-1}$ for both $c \in \{0,1\}$.
2. Compute $\mathsf{ct}^{0,i}_c := \mathsf{k}^i_c \oplus s_{c,i}$ and $\mathsf{ct}^{1,i}_c := \mathsf{k}^i_{1\oplus c} \oplus s_{c,i}$ for all $i \in [m], c \in \{0,1\}$.
3. Generate $\mathsf{ot}^0_{2,i} \leftarrow \mathsf{OT}'_2(\mathsf{ot}'_{1,i}, (\mathsf{k}^i_0, \mathsf{k}^i_1))$ and $\mathsf{ot}^1_{2,i} \leftarrow \mathsf{OT}'_2(\mathsf{ot}_{1,i}, ((\mathsf{ct}^{0,i}_0, \mathsf{ct}^{0,i}_1), (\mathsf{ct}^{1,i}_0, \mathsf{ct}^{1,i}_1)))$ for all $i \in [m]$.
4. Sample $\mathcal{I} \leftarrow \{0,1\}^m$.
5. Send $\{\mathsf{ot}^0_{2,i}, \mathsf{ot}^1_{2,i}\}_{i\in[m]}, \mathcal{I}$ to R.

Round 3 (Receiver).
1. For all $i \in [m]$ if $\mathcal{I}_i = 0$ compute $\mathsf{ot}^0_{3,i} \leftarrow \mathsf{OT}'_3(\mathsf{ot}^0_{1,i}, \mathsf{ot}^0_{2,i})$; otherwise compute $\mathsf{ot}^1_{3,i} \leftarrow \mathsf{OT}'_3(\mathsf{ot}^1_{1,i}, \mathsf{ot}^1_{2,i})$.
2. Send $\{\mathsf{ot}^{\mathcal{I}_i}_{3,i}\}_{i\in[m]}$ to S.

Offline computation.

Sender:
1. For all $i \in [m]$ if $\mathcal{I}_i = 0$ check that $\bot \neq \mathsf{OT}'^S_4(\mathsf{ot}^0_{1,i}, \mathsf{ot}^0_{2,i}, \mathsf{ot}^0_{3,i})$ otherwise check that $\bot \neq \mathsf{OT}'^S_4(\mathsf{ot}^1_{1,i}, \mathsf{ot}^1_{2,i}, \mathsf{ot}^1_{3,i})$.
2. If the previous checks do not succeed output $\bot$; otherwise output s_0, s_1.

Receiver:
1. Obtain $\mathsf{k}^i_{b_i} := \mathsf{OT}'^R_4(\mathsf{ot}^0_{1,i}, \mathsf{ot}^0_{2,i})$ and $(\mathsf{ct}^{d_i,i}_0, \mathsf{ct}^{d_i,i}_1) := \mathsf{OT}'^R_4(\mathsf{ot}^1_{1,i}, \mathsf{ot}^1_{2,i})$ for all $i \in [m]$.
2. Compute $s'_{b,i} := \mathsf{k}^i_{b_i} \oplus \mathsf{ct}^{d_i,i}_b$ for all $i \in [m]$ and output $s'_b := \bigoplus_{i\in[m]} s'_{b,i}$.

The correctness of the described protocol follows directly from the correctness of the underlying building blocks using the same arguments as given in OT'.

Now, we sate the security theorem for our OT protocol $\mathsf{OT}^{\mathsf{ListRew}}$.

Theorem 5.1. *Let X be a high min-entropy random variable defined by a probability distribution $\mathcal{D}$ and let OT' be the three-round OT protocol described in Figure 4.1 for the functionality $\mathcal{F}^{\mathcal{U}}_{\mathsf{OT}}$ (where $\mathcal{U}$ is the uniform distribution), then the OT protocol $\mathsf{OT}^{\mathsf{ListRew}}$ described in Figure 5.1 is a three-round sender list simulatable OT against a sometimes aborting receivers for the functionality $\mathcal{F}^{\mathcal{D}}_{\mathsf{OT}}$ (according to Definition 3.2) otherwise, $\mathsf{OT}^{\mathsf{ListRew}}$ is sender private (according to Definition 2.6). Moreover $\mathsf{OT}^{\mathsf{ListRew}}$ is a B-rewind receiver private OT (according to Definition 3.5), with $B = 2$. $\mathsf{OT}^{\mathsf{ListRew}}$ makes black-box use of OT'.*

We split the theorem into three lemmas, the first two focusing on the simulatability (Lemma 5.2) and privacy (Lemma 5.7) against a malicious receiver and the third focusing on the privacy of the receiver against a malicious sender (Lemma 5.8). The first two lemmas are stated and proven in Section 5.1, whereas the third is proven in Section 5.2.

5.1 List Simulatability Against Malicious Receivers

We prove this theorem by splitting it into two lemmas:

Lemma 5.2. *Let X be a high min-entropy random variable defined by a probability distribution $\mathcal{D}$ and let OT' be the three-round OT protocol described in Figure 4.1 for the functionality $\mathcal{F}_{\mathsf{OT}}^{\mathcal{U}}$ (where $\mathcal{U}$ is the uniform distribution), then the OT protocol $\mathsf{OT}^{\mathsf{ListRew}}$ described in Figure 5.1 is a three-round sender list simulatable OT against a non-aborting receiver for the functionality $\mathcal{F}_{\mathsf{OT}}^{\mathcal{D}}$ (accordingly to Definition 3.2).*

We prove that the OT protocol OT' described in Figure 5.1 is sender list simulatable by showing that a malicious non-aborting receiver R^* is not able to distinguish between a real execution of the protocol and an ideal execution using the simulator $\mathsf{Sim} = (\mathsf{Sim}^1, \mathsf{Sim}^2)$ described in Figure 5.2. Sim parametrized by the distribution $\mathcal{D}$ will make use of the simulator Sim' of OT' defined in Figure 4.2 parametrized by the distribution $\mathcal{D}$. We prove this indistinguishability using a sequence of hybrids that we describe below:

Hybrid H_0: This hybrid corresponds to the real world.

Hybrid H_1: This hybrid is almost identical to the previous hybrid with the only difference that the input bit b of the receiver is extracted. In order to do so the hybrid follows the steps (1), (2) and (3) of Sim described in Figure 5.2, the hybrid outputs the view of R^* in the first thread. The indistinguishability of the hybrids follows with similar arguments to the one described in the Claims 5.3 & 5.4.

Hybrid H_1': In this hybrid, we proceed as in the previous hybrid except that, after the extraction of the input, the hybrid proceeds as explained in steps (4) and (6) of Sim described in Figure 5.2, with the difference that the second round is computed honestly (as in steps (2) and (3)). The hybrid outputs the same output as Sim. The indistinguishability of the hybrids follows with similar arguments to the one described in the Claims 5.3 & 5.4.

Hybrid $\mathsf{H}_{2,0}^x$: In this hybrid, we proceed as in the previous hybrid but in step (x) the message ot_{2,j^*}^0 is generated as described in step (4).g of Figure 5.2 (i.e. using the simulator $\mathsf{Sim}_{j^*,0}^2$ w.r.t. input $\mathsf{k}_{b_{j^*}}^{j^*} \leftarrow \{0,1\}^\lambda$). For the remaining indices $i \in [m] \setminus \{j^*\}$ both of the keys remain randomly sampled $\mathsf{k}_0^i, \mathsf{k}_1^i \leftarrow \{0,1\}^\lambda$. The indistinguishability of the hybrids H_1 and H_2 follows from the sender privacy of the underlying OT protocol OT', which we formally prove in Claim 5.5.

Hybrid H_3^x: In this hybrid, in step (x) we switch from an encryption $\mathsf{ct}_{1\oplus b}^{d_{j^*},j^*} := \mathsf{k}_{1\oplus b_{j^*}}^{j^*} \oplus s_{1\oplus b,j^*}$ to a randomly sampled ciphertext $\mathsf{ct}_{1\oplus b}^{d_{j^*},j^*} \leftarrow \{0,1\}^\lambda$ (to be more precise, x is used to denote the 4th or the 6th stage of the simulation, namely $x = 4$ or $x = 6$). For the remaining indices $i \in [m] \setminus \{j^*\}$ all the ciphertexts remain generated as described in the protocol. Since, due to the previous hybrid, the key $\mathsf{k}_{1\oplus b_{j^*}}^{j^*}$ is not part of the execution anymore, we can rely on the perfect security of the one-time pad to show that hybrids are perfectly indistinguishable, the proof follows similarly to the one described in Claim 4.7.

Hybrid $\mathsf{H}_{2,1}^x$: In this hybrid, in step (x) we proceed as in the previous hybrid but the message ot_{2,j^*}^1 is generated as described in step (4).h (to be more precise, x is used to denote the 4th or the 6th stage of the simulation, namely $x = 4$ or $x = 6$). In other words, ot_{2,j^*}^1 is generated using the simulator $\mathsf{Sim}_{j^*,1}^2$ w.r.t. messages $\mathsf{ct}_b^{d_{j^*},j^*} := \mathsf{k}_{d_{j^*}\oplus b_{j^*}}^{j^*} \oplus s_{b,j^*}$, $\mathsf{ct}_{1\oplus b}^{d_{j^*},j^*} \leftarrow \{0,1\}^\lambda$. For the remaining indices $i \in [m] \setminus \{j^*\}$ all of the ciphertexts remain correctly generated, i.e. $\mathsf{ct}_c^{0,i} := \mathsf{k}_c^i \oplus s_{c,i}$ and $\mathsf{ct}_c^{1,i} := \mathsf{k}_{1\oplus c}^i \oplus s_{c,i}$ for $c \in \{0,1\}$. The indistinguishability of the hybrids follows from the list simulatable sender security of the underlying OT protocol OT', which we formally prove in Claim 5.6.

Hybrid H_4^x: In this hybrid, in step (x) we switch from an n-out-of-n secret sharing of the input s_{1-b} to an n-out-of-n secret sharing of a random value r (to be more precise, x is used to denote the 4th or the 6th stage of the simulation, namely $x = 4$ or $x = 6$). The secret shares $s_{1-b,1}, \ldots, s_{1-b,j^*-1}, s_{1-b,j^*+1}, \ldots, s_{1-b,m}$ are then used as in the previous hybrid for the generation of the ciphertexts $\mathsf{ct}_c^{0,i}, \mathsf{ct}_c^{1,i}$ for all $i \in [m] \setminus \{j^*\}, c \in \{0,1\}$. The perfect indistinguishability between the hybrids follows from the security of the n-out-of-n secret sharing scheme, the proof follows similar to the proof of Claim 4.8

Hybrid H_5: This hybrid corresponds to the simulator described in Figure 5.2.

From the above arguments we can conclude that $\mathsf{H}_0 \approx \mathsf{H}_1 \approx \mathsf{H}'_1 \approx_c \mathsf{H}^4_{2,0} \approx_c \mathsf{H}^4_3 \approx_c \mathsf{H}^4_{2,1} \approx \mathsf{H}^4_4 \approx_c \mathsf{H}^6_{2,0} \approx_c \mathsf{H}^6_3 \approx_c \mathsf{H}^6_{2,1} \approx \mathsf{H}^6_4 \approx \mathsf{H}_5$.

Figure 5.2: The simulator $\mathsf{Sim} = (\mathsf{Sim}^1, \mathsf{Sim}^2)$ for $\mathsf{OT}^{\mathsf{ListRew}}$.

Let $(\mathsf{Sim}'^1_{i,b}, \mathsf{Sim}'^2_{i,b})$ be the simulator defined in Figure 4.2 parametrized by the uniform distribution $\mathcal{U}$ for one of the two OT' executions in position i, specifically if $b = 0$ we indicate the left execution, i.e. the messages ot generated in OT', and the right execution, i.e. the messages ot' generated in OT', otherwise.
Input. The simulator Sim^1 takes as input the distribution $\mathcal{D}$ and 1^λ.

1) First thread: Sim^1 computes the following steps:

1. For all $i \in [m]$ and $b \in \{0,1\}$ upon receiving $\{\mathsf{ot}^0_{1,i}, \mathsf{ot}^1_{1,i}\}_{i\in[m]}$ from R^* generates $\mathsf{ot}^b_{2,i}$ as the honest sender would do. Moreover, sample $\mathcal{I} \in \{0,1\}^m$ at random and send $\{\mathsf{ot}^0_{2,i}, \mathsf{ot}^1_{2,i}\}_{i\in[m]}$ to R^*.
2. If R^* does not send a 3rd round, abort and output the view of R^*.
3. Upon receiving $\{\mathsf{ot}^{\mathcal{I}_i}_{3,i}\}_{i\in[m]}$ from R^*, initialize $\mathsf{Defense}$ with the set of defenses of $\{\mathsf{ot}^{\mathcal{I}_i}_{1,i}\}_{i\in[m]}$ obtained from $\{\mathsf{ot}^{\mathcal{I}_i}_{3,i}\}_{i\in[m]}$ (the obtained defenses are defenses for the underlying oblivious transfer OT').

2) Estimate the abort probability: Sim^1 initializes $\mathsf{ctr} := 0, T := 0$ and computes the following steps:

1. Increment T, i.e. $T := T + 1$.
2. Compute and send a new 2nd round with fresh randomness to R^* as described in step (1).1 (i.e. as an honest sender would do). If a valid 3rd round is received from R^* increment ctr, i.e. $\mathsf{ctr} := \mathsf{ctr} + 1$.
3. If $\mathsf{ctr} = 12\lambda$ then output $p = \frac{12\lambda}{T}$, otherwise, perform a new rewind, i.e. go back to step (2).1.
4. Set $\mathsf{maxrew} = \lceil \frac{\lambda}{p} \rceil$.

3) Extracting the input of R^*: Sim^1 initializes $\mathsf{ctr} := 0$ and computes the following steps:

1. Perform the rewinds as follows:

   Rewinding Threads: Sim computes the following steps:

   (a) Increment ctr, i.e. $\mathsf{ctr} := \mathsf{ctr} + 1$.
   (b) Compute and send a new 2nd round with fresh randomness to R^* as described in step (1).1 (i.e. as an honest sender would do) and upon receiving $\{\widetilde{\mathsf{ot}}^{\mathcal{I}_i}_{3,i}\}_{i\in[m]}$ from R^* check that the defenses $\widetilde{\mathsf{Defense}}$ contained in $\{\widetilde{\mathsf{ot}}^{\mathcal{I}_i}_{3,i}\}_{i\in[m]}$ are valid w.r.t. $\{\mathsf{ot}^{\mathcal{I}_i}_{1,i}\}_{i\in[m]}$. Finally, set $\mathsf{Defense} := \mathsf{Defense} \cap \widetilde{\mathsf{Defense}}$.
   (c) If $\mathsf{ctr} > \mathsf{maxrew}$ output fail.
   (d) Check that for at least one slot j^* all the defenses of $\mathsf{ot}^0_{1,j^*}, \mathsf{ot}^1_{1,j^*}$ are in $\mathsf{Defense}$. If this is the case, proceed to next step; else continue the rewinding, i.e. return to step (3).1.

2. Follow step (3).1 of $(\mathsf{Sim}'^1_{j^*,0}, \mathsf{Sim}'^1_{j^*,1})$ and use the defenses contained in $\mathsf{Defense}$ to extract the bits b_{j^*} and d_{j^*} used by R^* in $\mathsf{ot}^0_{1,j^*}, \mathsf{ot}^1_{1,j^*}$, let $b = b_{j^*} \oplus d_{j^*}$.

4) Estimate the abort probability: Sim^1 initializes $\mathsf{ctr} := 0, T := 0$ and computes the following steps:

1. Increment T, i.e. $T := T + 1$.
2. Compute a new 2nd round with fresh randomness and compute $\{\mathsf{ot}^0_{2,i}, \mathsf{ot}^1_{2,i}\}_{i\in[m]}$ as follows:

   (a) Sample s_b, r at random from the distribution $\mathcal{D}$.
   (b) Sample $\mathsf{k}^{j^*}_{b_{j^*}} \leftarrow \{0,1\}^\lambda$ and sample $\mathsf{k}^i_0, \mathsf{k}^i_1 \leftarrow \{0,1\}^\lambda$ for all $i \in [m] \setminus \{j^*\}$.
   (c) Sample $s_{b,l} \leftarrow \{0,1\}^\lambda$ for all $l \in [m] \setminus \{j^*\}$ and set $s_{b,j^*} := s_b \oplus s_{b,1} \oplus \cdots \oplus s_{b,m}$.
   (d) Sample $s_{1-b,l} \leftarrow \{0,1\}^\lambda$ for all $l \in [m] \setminus \{j^*\}$ and set $s_{1-b,j^*} := r \oplus s_{1-b,1} \oplus \cdots \oplus s_{1-b,m}$.
   (e) Set $\mathsf{ct}^{d_{j^*},j^*}_b := \mathsf{k}^{j^*}_{d_{j^*} \oplus b_{j^*}} \oplus s_{b,j^*}$, $\mathsf{ct}^{d_{j^*},j^*}_{1 \oplus b} \leftarrow \{0,1\}^\lambda$.
   (f) For all $i \in [m] \setminus \{j^*\}$, set $\mathsf{ct}^{0,i}_c := \mathsf{k}^i_c \oplus s_{c,i}$ and $\mathsf{ct}^{1,i}_c := \mathsf{k}^i_{1\oplus c} \oplus s_{c,i}$ for all $i \in [m] \setminus \{j^*\}, c \in \{0,1\}$.
   (g) For all $i \in [m] \setminus \{j^*\}$ generate $\mathsf{ot}^0_{2,i} \leftarrow \mathsf{OT}'_2(\mathsf{ot}'_{1,i}, (\mathsf{k}^i_0, \mathsf{k}^i_1))$ and $\mathsf{ot}^1_{2,i} \leftarrow \mathsf{OT}'_2(\mathsf{ot}_{1,i}, ((\mathsf{ct}^{0,i}_0, \mathsf{ct}^{0,i}_1), (\mathsf{ct}^{1,i}_0, \mathsf{ct}^{1,i}_1)))$.
   (h) Generate ot^0_{2,j^*} as defined in steps (6.2) of $\mathsf{Sim}'^2_{j^*,0}$ w.r.t. input $\mathsf{k}^{j^*}_{b_{j^*}}$.
   (i) Generate ot^1_{2,j^*} as defined in steps (6.2) of $\mathsf{Sim}'^2_{j^*,1}$ w.r.t. input $\mathsf{ct}^{d_{j^*},j^*}_b$.

   If a valid 3rd round is received from R^* increment ctr, i.e. $\mathsf{ctr} := \mathsf{ctr} + 1$.
3. If $\mathsf{ctr} = 12\lambda$ then output $q = \frac{12\lambda}{T}$, otherwise, perform a new rewind, i.e. return to step (4).1.
4. Set $\mathsf{maxrew}' = \lceil \frac{\lambda}{q} \rceil$.

5) Query the ideal functionality: Sim^1 sends to ideal functionality $(b_{j^*}, 1^d)$ receiving $(s^1_{b_{j^*}}, \ldots, s^d_{b_{j^*}})$, with $d := \lambda \cdot \mathsf{maxrew}'$.

6) Forcing the output Sim^2 on input $(s^1_{b_{j^*}}, \ldots, s^d_{b_{j^*}})$ initializes $\mathsf{ctr} = 0$ and computes the following steps:

Threads to force the output:

1. Increment ctr, i.e. $\mathsf{ctr} := \mathsf{ctr} + 1$
2. Compute $\{\mathsf{ot}^0_{2,i}, \mathsf{ot}^1_{2,i}\}_{i\in[m]}$ as described in step 4 "Estimate the abort probability" but using $s^{\mathsf{ctr}}_{b_{j^*}}$ as input.
3. If $\mathsf{ctr} > \lambda \cdot \mathsf{maxrew}'$ output fail.
4. If R^* outputs an accepting third round, then output the view of R^* in this thread; else continue the rewinding, i.e. return to step (6).1.

Sim also keeps a count of its overall running time and if it reaches 2^λ steps it outputs fail.

Claim 5.3 Sim *runs in expected polynomial time in* λ*.*

Proof. The analysis of the simulator follows similarly to the analysis in [GK96] as recapped in [Lin16]. We analyze the running time in more detail. Let ε be the probability that the adversary R^* outputs a valid third round. The following analysis takes into account that each time Sim follows the steps of $\{\mathsf{Sim}'_{i,b}\}_{i\in[m],b\in\{0,1\}}$ it runs in polynomial time.

In the first step ("First thread") the simulator clearly runs in polynomial time. The estimation of the probability for the second step ("Estimate the abort probability") is executed until Sim does not collect 12λ valid third rounds from R^*. Since the probability that R^* outputs, a valid third round is ε, Sim stops after $12\lambda/\varepsilon$ attempts.

From the analysis in [GK96] it follows that the estimation p is within a constant factor from ε, unless of a negligible probability $2^{-\lambda}$ (therefore the case of running 2^λ steps adds only a polynomial amount of overhead to the simulator).

In the third step ("Extracting the input of R^*") the simulator Sim stops after $\lambda \cdot \mathsf{maxrew}$, which is $\frac{\mathrm{poly}(\lambda)}{p}$, iterations. The fifth step ("Query the ideal functionality") happens, as the first step, in polynomial time. Relying on the same arguments as in the previous steps, one can also argue that the fourth and six steps are expected polynomial time.

This results in an overall running time of the simulator that is expected polynomial in λ. □

Claim 5.4 Sim *outputs* fail *with negligible probability in* λ*.*

Proof. There are three possibilities that the simulator Sim outputs fail:

1. The simulator executes more than 2^λ steps
2. The simulator executes more than maxrew rewinds in step 3 ("Extracting the input of R^*")
3. The simulator executes more than $\lambda \cdot \mathsf{maxrew}'$ rewinds in step 6 ("Forcing the output")

The first case is directly bounded by the previous proof, i.e. if the simulator runs in expected polynomial time in λ then it only executes more than 2^λ steps with negligible probability.

To show that the second case is negligible, we need to bound the probability with which Sim extracts the input of R^*.

From the (simulation) security of the underlying OT' follows that Sim is able to extract if for the two executions in a position j^* of OT' the simulator Sim collects all the defenses necessary for ot^b_{1,j^*} (which are two) with $b \in \{0,1\}$. Indeed, Sim once she collects all the defense for ot^b_{1,j^*} she can invoke the underlying simulator $\mathsf{Sim}'^{,2}_{j^*,b}$ of OT', which on inputs defense for ot^b_{1,j^*} extracts the input bit contained in ot^b_{1,j^*}, with $b \in \{0,1\}$.

Due to the above arguments we can conclude that Sim is able to extract successfully if 4 different valid defenses for $\mathsf{ot}^0_{1,j^*}, \mathsf{ot}^1_{1,j^*}$ for a single index $j^* \in [m]$ are obtained. The simulator executes each rewind with new fresh randomness, i.e. the set $\mathcal{I}$ is always randomly sampled, it follows that the simulator Sim overall (in the worst case) collects maxrew third rounds from R^*. Since $\mathcal{I}$ is sampled freshly with every rewind, it follows that the probability that the simulator is not able to extract for at least a single index is $1 - \left(\frac{1}{4^m}\right)^{\mathsf{maxrew}}$, which is the same probability that, for a single index, the simulator does not succeed if one of the four cases, $00, 01, 10, 11$, never occurs, i.e. $\mathcal{I}$ never contains these indices taken this over all the m indexes. Therefore, the success probability of the simulator is equal to $1 - \left(\frac{1}{4^m}\right)^{\mathsf{maxrew}}$ Therefore, Sim successfully extracts except with negligible probability.

To bound the probability of the third case we first prove that the probability that R^* replies with a valid third round to a simulated 2nd round (as described in step (4)) is negligible close to the probability that R^* replies with a valid third round to an honestly computed 2nd round of the protocol.

In order to do so we consider the following hybrids:

Hybrid H_0: We start by considering an hybrid were the messages of the sender are computed honestly but the input bits b_{j^*} and d_{j^*} of the receiver's messages $\mathsf{ot}^0_{1,j^*}, \mathsf{ot}^1_{1,j^*}$ are extracted in exponential time to compute $b = d_{j^*} \oplus b_{j^*}$. The rest of the protocol is executed as the honest sender.

Hybrid H_1: In this hybrid, we proceed as in H_0 but the message ot^0_{2,j^*} is generated as described in step (4).g (i.e. using the simulator $\mathsf{Sim}'^2_{j^*,0}$ w.r.t. input $\mathsf{k}^{j^*}_{b_{j^*}} \leftarrow \{0,1\}^\lambda$). For the remaining indices $i \in [m] \setminus \{j^*\}$ both of the keys remain randomly sampled $\mathsf{k}^i_0, \mathsf{k}^i_1 \leftarrow \{0,1\}^\lambda$. The indistinguishability of the hybrids follows from the list simulatable sender security of the underlying OT protocol OT', the proof follows similar to the one described in Claim 5.5.

Hybrid H_2: In this hybrid, we switch from an encryption $\mathsf{ct}^{d_{j^*},j^*}_{1\oplus b} := \mathsf{k}^{j^*}_{1\oplus b_{j^*}} \oplus s_{1\oplus b,j^*}$ to a randomly sampled ciphertext $\mathsf{ct}^{d_{j^*},j^*}_{1\oplus b} \leftarrow \{0,1\}^\lambda$. For the remaining indices $i \in [m] \setminus \{j^*\}$ all the ciphertexts remain generated as described in the protocol. Since, due to the previous hybrid, the key $\mathsf{k}^{j^*}_{1\oplus b_{j^*}}$ is not part of the execution anymore, we can rely on the perfect security of the one-time pad to show that hybrids are perfectly indistinguishable. The proof follows similar to the one described in Claim 4.7.

Hybrid H_3: In this hybrid, we proceed as in H_2 but the message ot^1_{2,j^*} is generated as described in step (4).h (i.e. using the simulator $\mathsf{Sim}'^2_{j^*,1}$ w.r.t. input $\mathsf{ct}^{d_{j^*},j^*}_b := \mathsf{k}^{j^*}_{d_{j^*}\oplus b_{j^*}} \oplus s_{b,j^*}$, $\mathsf{ct}^{d_{j^*},j^*}_{1\oplus b} \leftarrow \{0,1\}^\lambda$). For the remaining indices $i \in [m] \setminus \{j^*\}$ all of the ciphertexts remain correctly generated, i.e. $\mathsf{ct}^{0,i}_c := \mathsf{k}^i_c \oplus s_{c,i}$ and $\mathsf{ct}^{1,i}_c := \mathsf{k}^i_{1\oplus c} \oplus s_{c,i}$ for $c \in \{0,1\}$. The indistinguishability of the hybrids follows from the list simulatable sender security of the underlying OT protocol OT'. The proof follows similar to the one described in Claim 5.6.

Hybrid H_4: In this hybrid, we switch from an n-out-of-n secret sharing of the input s_{1-b} to an n-out-of-n secret sharing sharing of a random value r. The secret shares $s_{1-b,1}, \ldots, s_{1-b,j^*-1}, s_{1-b,j^*+1}, \ldots, s_{1-b,m}$ are then used as in the previous hybrid for the generation of the ciphertexts $\mathsf{ct}^{0,i}_c, \mathsf{ct}^{1,i}_c$ for all $i \in [m] \setminus \{j^*\}, c \in \{0,1\}$. The perfect indistinguishability between this and the previous hybrid follows from the security of the n-out-of-n secret sharing scheme. The proof follows similar to the one described in Claim 4.8.

From the above hybrids, we can conclude that the probability that R^* replies with a valid third round to an honestly computed 2nd round, which we denote with ε, is negligible close to the probability that R^* replies with a valid third round to simulated 2nd round (as described in step (4)). Therefore, the probability that R^* replies with a valid third round after receiving a simulated 2nd round is $\varepsilon - \nu(\lambda)$, where $\nu(\lambda)$ is a negligible function. Since we are considering a non-aborting R^* where ε is non-negligible, we can assume that $\varepsilon > 2\nu(\lambda)$. Following the analysis of [Lin16, Claim 5.8 case 2], we can conclude that the simulator executes more than $\lambda \cdot \mathsf{maxrew}'$ rewinds in step 6 only with negligible probability. □

Claim 5.5 *Let OT' be the three-round sender list-simulatable OT protocol, then H_1 and H_2 are computationally indistinguishable.*

Proof. Suppose by contradiction that this is not the case, then there exists an index $j^* \in [m]$ s.t. the mentioned hybrids are distinguishable. Let CH be the challenger of the sender private OT'.

We fix the first round message $\{\mathsf{ot}^0_{1,i}, \mathsf{ot}^1_{1,i}\}_{i\in[m]}$ that maximizes the distinguishing advantage of the adversary R^* in the two above experiments. Note that in OT' the inputs of the receiver are fixed in the first round, therefore they are fixed once $\{\mathsf{ot}^0_{1,i}, \mathsf{ot}^1_{1,i}\}_{i\in[m]}$ have been generated. Let d_{j^*}, b_{j^*} be the inputs of the receiver used in $\mathsf{ot}^0_{1,j^*}, \mathsf{ot}^1_{1,j^*}$, respectively. We define the adversary $\mathcal{A}$ of OT' which internally runs R^* and acts as a proxy between R^* and the challenger of OT'.

In more detail, $\mathcal{A}$ receives as an auxiliary input $(\{\mathsf{ot}^0_{1,i}, \mathsf{ot}^1_{1,i}\}_{i\in[m]},\ d_{j^*}, b_{j^*})$ and behaves as follows: it sets $b = d_{j^*} \oplus b_{j^*}$ and sends ot^0_{1,j^*} as well as $\mathsf{k}^{j^*}_{b_{j^*}}, \mathsf{k}'^{j^*}_{1-b_{j^*}}$ to the challenger (where $\mathsf{k}^{j^*}_{b_{j^*}}, \mathsf{k}'^{j^*}_{1-b_{j^*}}$ are randomly sampled). Upon receiving ot^0_{2,j^*} from the challenger, $\mathcal{A}$ does the following:

1. Compute the second-round messages $\{\mathsf{ot}^0_{2,j}, \mathsf{ot}^1_{2,j}\}_{j\in[m], j\neq j^*}$ as the honest sender would do.
2. Compute $\{\mathsf{ct}^{d_{j^*},j^*}_c, \mathsf{ct}^{1-d_{j^*},j^*}_c\}_{c\in\{0,1\}}$ as the as the honest sender would do, but using the keys $\mathsf{k}^{j^*}_{b_{j^*}}$ and $\mathsf{k}'^{j^*}_{1-b_{j^*}}$.
3. Compute ot^1_{2,j^*} w.r.t. messages $\{\mathsf{ct}^{d_{j^*},j^*}_c, \mathsf{ct}^{1-d_{j^*},j^*}_c\}_{c\in\{0,1\}}$ as the honest sender would do.
4. Send $\{\mathsf{ot}^0_{2,i}, \mathsf{ot}^1_{2,i}\}_{i\in[m]}$ to R^* and output the view of R^*.

The reduction runs the distinguisher (which exists by contradiction) on the output of $\mathcal{A}$. The reduction outputs whatever the distinguisher outputs. □

Claim 5.6 *Let* OT' *be the three-round list-simulatable OT protocol, then* H_2 *and* H_3 *are computationally indistinguishable.*

Proof. Suppose by contradiction that this is not the case, then there exists an index $j^* \in [m]$ s.t. the above described hybrids are distinguishable. Let CH be the challenger of the sender private OT'.

We fix the first round message $\{\mathsf{ot}^0_{1,i}, \mathsf{ot}^1_{1,i}\}_{i\in[m]}$ that maximizes the distinguishing advantage of the adversary R^* in the two above experiments. Note that in OT' the inputs of the receiver are fixed in the first round, therefore they are fixed once $\{\mathsf{ot}^0_{1,i}, \mathsf{ot}^1_{1,i}\}_{i\in[m]}$ are generated. Let d_{j^*}, b_{j^*} be the inputs of the receiver used in $\mathsf{ot}^0_{1,j^*}, \mathsf{ot}^1_{1,j^*}$, respectively. We define the adversary $\mathcal{A}$ of OT' which internally runs R^* and acts as a proxy between R^* and the challenger of OT'.

In more detail, $\mathcal{A}$ receives as an auxiliary input $(\{\mathsf{ot}^0_{1,i}, \mathsf{ot}^1_{1,i}\}_{i\in[m]}, d_{j^*}, b_{j^*})$ and behaves as follows: it sets $b = d_{j^*} \oplus b_{j^*}$ and sends ot^0_{1,j^*} and $\{\mathsf{ct}^{d_{j^*},j^*}_c, \mathsf{ct}^{1-d_{j^*},j^*}_c\}_{c\in\{0,1\}}$ to the challenger (where $\{\mathsf{ct}^{d_{j^*},j^*}_c, \mathsf{ct}^{1-d_{j^*},j^*}_c\}_{c\in\{0,1\}}$ are randomly sampled). Upon receiving ot^1_{2,j^*} from the challenger, $\mathcal{A}$ does the following.

1. Compute the second-round messages $\{\mathsf{ot}^0_{2,j}, \mathsf{ot}^1_{2,j}\}_{j\in[m], j\neq j^*}$ as the honest sender would do.
2. Compute s_{b,j^*} as described by the simulator (see Figure 5.2).
3. Compute ot^0_{2,j^*} as described by the simulator (see Figure 5.2) w.r.t. the key $\mathsf{k}^{j^*}_{b_{j^*}} = ct^{d_{j^*},j^*}_b \oplus s_{b,j^*}$.
4. Send $\{\mathsf{ot}^0_{2,i}, \mathsf{ot}^1_{2,i}\}_{i\in[m]}$ to R^* and output the view of R^*.

The reduction runs the distinguisher (which exists by contradiction) on the output of $\mathcal{A}$. The reduction outputs whatever the distinguisher outputs. □

Lemma 5.7. *Let* OT' *be the three-round OT protocol described in Figure 4.1, then the OT protocol* $\mathsf{OT}^{\mathsf{ListRew}}$ *described in Figure 5.1 is is sender private (accordingly to Definition 2.6).*

The proof for sender privacy against an aborting R^* for $\mathsf{OT}^{\mathsf{ListRew}}$ proceeds in the same way as the proof for sender privacy against an aborting R^* for OT' (Lemma 4.9), with the difference that it is now necessary to rely on OT' instead of OT.

5.2 *B*-Rewind Security

In this section, we state and prove the second part of Theorem 5.1, Namely, the B-rewind security for the receiver.

Lemma 5.8. OT' *be the three-round OT protocol described in Figure 4.1 then* $\mathsf{OT}^{\mathsf{ListRew}}$ *is a B-rewind receiver private OT (accordingly to Definition 3.5), with $B = 3$.*

Proof. We prove the receiver indistinguishability by relying on a sequence of hybrids:

H_0**:** This hybrid corresponds to the experiment E^0 (i.e. where the adversary may send B-second rounds and expect B-third rounds from the receiver) where the bit 0 is used as the input of the receiver.

H_k**:** This hybrid corresponds to the previous hybrid where in the first k executions of OT the bits $d_i = 1 \oplus b_i$ instead of $d_i = b_i$ are used.

H_m**:** This hybrid corresponds to the experiment E^1, where 1 is used as the input of the receiver.

In Claim 5.9, we show that $\mathsf{H}_{k-1} \approx \mathsf{H}_k$ for all $k \in [m]$, which concludes the proof of the lemma. □

Claim 5.9 *The hybrids H_{k-1} and H_k are computationally indistinguishable for any $k \in [m]$.*

Proof. For simplicity, we show that H_0 is computationally indistinguishable from H_1. The argument we use carries over straightforwardly to the general case.

We prove this claim by contradiction. We assume that there exists an adversary $\mathcal{A}'$, that can distinguish between the two hybrids H_0 and H_1 with non-negligible probability, then we can use $\mathcal{A}'$ to construct an adversary $\mathcal{A}$ that breaks the receiver privacy of the OT protocol OT (used to build OT' which is used to build $\mathsf{OT}^{\mathsf{ListRew}}$). Let CH be the challenger for the receiver private OT. $\mathcal{A}$ will be interacting as the malicious sender w.r.t. CH in the execution of OT, and as a receiver with $\mathcal{A}$.

$\mathcal{A}$ first makes a random guess of which executions are not queried by $\mathcal{A}'$ in the execution of $\mathsf{OT}^{\mathsf{ListRew}}$. We recall that $\mathsf{OT}^{\mathsf{ListRew}}$ is composed of $2m$ executions of OT', therefore, for the first position, it holds that there are two executions of OT', which we denote as $\mathsf{ot}^0_{1,1}$ or $\mathsf{ot}^1_{1,1}$, of which one will be completed based on the bit of the sender, which is represented by $\mathcal{A}'$. Note that in the construction of OT' in each position two oblivious transfer protocols OT are executed of which the receiver will finish one of them. The finished execution depends on the bit chosen by the sender the sender, i.e. $\mathcal{A}'$. We denote the messages that are part of $\mathsf{ot}^{\mathcal{I}_1}_{1,1}$ (i.e. the message of the underlying OT) as $\overline{\mathsf{ot}}^{\mathcal{I}_1,0}_{1,1}$, $\overline{\mathsf{ot}}^{\mathcal{I}_1,1}_{1,1}$. This results in the fact that there are actually two indices that need to be guessed here. The first index, $\mathcal{I}_1$, is the index of the outer protocol that decides if $\mathsf{ot}^0_{1,1}$ is being continued, which happens in case that $\mathcal{I}_1 = 0$, or if $\mathsf{ot}^1_{1,1}$ is being continued, which happens in the case that $\mathcal{I}_1 = 1$. Furthermore, in the underlying protocol OT', another index $\mathcal{I}'_1$ is guessed which decides if either the left execution of OT, in the case that $\mathcal{I}'_1 = 0$, or the right execution of OT, in the case that $\mathcal{I}'_1 = 1$, is continued. Therefore, the adversary $\mathcal{A}$ needs to samples two random bits where the first is the guess of $\mathcal{I}_1$ and the second is the guess of $\mathcal{I}'_1$.

$\mathcal{A}$ then proceeds as follows:

1. $\mathcal{A}$ sets $\neg\mathcal{I}_1 = 1 \oplus \mathcal{I}_1$ and samples two indices $\mathcal{I}_1$, $\mathcal{I}'_1$ and a random bit $b_1 \leftarrow \{0,1\}$. Compute $\mathsf{ot}^{\neg\mathcal{I}_1}_{1,1}$ as the honest receiver would do w.r.t. b_1 and compute $\mathsf{ot}^{\mathcal{I}_1}_{1,1}$ in the following way:
   (a) Sample a random bit $d_1^{\neg\mathcal{I}'_1} \leftarrow \{0,1\}$ and generate 1st round of OT (namely $\overline{\mathsf{ot}}^{\mathcal{I}_1,\neg\mathcal{I}'_1}_{1,1}$) w.r.t. $d_1^{\neg\mathcal{I}'_1}$. Set $d_1^{\neg\mathcal{I}'_1} = d_1^{\mathcal{I}'_1}$ and send to the challenger $(d_1^{\mathcal{I}'_1}, 1 \oplus d_1^{\mathcal{I}'_1})$ in order to obtain $\overline{\mathsf{ot}}^{\mathcal{I}_1,\mathcal{I}'_1}_{1,1}$.
2. Compute the remaining first rounds $\{\mathsf{ot}^0_{1,i}, \mathsf{ot}^1_{1,i}\}_{i\in[m]/\{1\}}$ as the sender would do and send $\{\mathsf{ot}^0_{1,i}, \mathsf{ot}^1_{1,i}\}_{i\in[m]}$ to $\mathcal{A}'$.
3. Upon receiving $\{\mathcal{I}^{\mathcal{A}'}_{1,j}\}_{j\in[B]}$ and $\{\mathsf{ot}^{0,j}_{2,i}, \mathsf{ot}^{1,j}_{2,i}\}_{i\in[m],j\in[B]}$, let $\mathcal{I}^{\mathcal{A}',b}_{i,j}$ denote the index contained in $\mathsf{ot}^{b,j}_{2,i}$. If there exists a $j \in [B]$ s.t. ($\mathcal{I}^{\mathcal{A}'}_{1,j} = \mathcal{I}_1$ and $\mathcal{I}_1^{\mathcal{A}',\mathcal{I}_1} = \mathcal{I}'_1$) abort and output a random bit, otherwise continue with the next step.
4. Compute $\{\mathsf{ot}^{\mathcal{I}^{\mathcal{A}'}_{i,j},j}_{3,i}\}_{i\in[m],j\in[B]}$ as the honest sender would do and output the view of $\mathcal{A}'$.

At this point, the reduction runs the distinguisher (which exists by contradiction) on the view of $\mathcal{A}'$ and outputs the output of the distinguisher.

We observe that if the challenger of the OT protocol OT generates the OT message $\overline{\mathsf{ot}}^{\mathcal{I}_1,\mathcal{I}'_1}_{1,1}$, using $d_1^{\mathcal{I}'_1}$, then this corresponds to hybrid H_0, otherwise, the reduction simulates hybrid H_1. Therefore if the reduction does not abort in step 4, the reduction distinguishes with probability $\frac{1}{2}+q$ where q is non-negligible. While if the reduction aborts in step 4 it samples a random bit. Since indices $\mathcal{I}_1$, $\mathcal{I}'_1$ are taken at random, the probability that the reduction does not abort is $\frac{1}{4}$. Therefore the winning probability is $((\frac{1}{2}+q)\frac{1}{4}+(1-\frac{1}{4})\frac{1}{2})\cdot p = (\frac{1}{2}+\frac{q}{4})\cdot p$, where p is the probability that $\mathcal{A}'$ sends a valid second round to $\mathcal{A}$. □

5.3 Parallel Repetition of $\mathsf{OT}^{\mathsf{ListRew}}$

Besides presenting the protocol on its own, we also need to show that a parallel repeated version of the protocol allows for the same type of a simulation-based notion. We denote the previous protocol as $\mathsf{OT}^{\mathsf{ListRew}} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3, \mathsf{OT}_4 = (\mathsf{OT}_4^S, \mathsf{OT}_4^R))$, where OT_4^S is the output computation computed by the sender and outputs either (s_0, s_1) or $\perp$ and OT_4^R is the output computation computed by the receiver and outputs either s_b or $\perp$. The parallel repeated version of the protocol can be found in Figure 5.3.

Figure 5.3: The parallel repetitions of $\mathsf{OT}^{\mathsf{ListRew}}$.

Initialization: S has in input the distribution $(s_0^i, s_1^i)_{i\in[n]}$. The receiver uses as its input n bits $b_1, \dots, b_n$. To not overburden the notation we set $n = \lambda$.

Round 1 (Receiver).

1. Generate $\mathsf{ot}_{1,i} := \mathsf{OT}_1(1^\lambda, b_i)$ for all $i \in [n]$.
2. Send $\{\mathsf{ot}_{1,i}\}_{i\in[n]}$ to S.

Round 2 (Sender).

1. Generate $\mathsf{ot}_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}_{1,i}, (s_0^i, s_1^i))$ for all $i \in [n]$.
2. Send $\{\mathsf{ot}_{2,i}\}_{i\in[n]}$ to R.

Round 3 (Receiver).

1. Compute $\mathsf{ot}_{3,i} := \mathsf{OT}_3(\mathsf{ot}_{1,i}, \mathsf{ot}_{2,i})$ for all $i \in [n]$.
2. Send $\{\mathsf{ot}_{3,i}\}_{i\in[n]}$ to S.

Offline computation.

Sender:

1. If $\perp = \mathsf{OT}_4^S(\mathsf{ot}_{1,i}, \mathsf{ot}_{2,i}, \mathsf{ot}_{3,i})$ for any $i \in [n]$ output $\perp$, else output $(s_0^i, s_1^i)_{i\in[n]}$.

Receiver:

1. If $\perp = \mathsf{OT}_4^R(\mathsf{ot}_{1,i}, \mathsf{ot}_{2,i})$ for any $i \in [n]$ output $\perp$, else output $(s_b'^i)_{i\in[n]}$ where $s_b'^i := \mathsf{OT}_4^R(\mathsf{ot}_{2,i})$.

Theorem 5.10. *Let X be a high min-entropy random variable defined by a probability distribution $\mathcal{D}$ and let $\mathsf{OT}^{\mathsf{ListRew}}$ the 1-out-of-two OT described in Figure 5.1 for the functionality $\mathcal{F}_{\mathsf{OT}}^{\mathcal{U}}$ (where $\mathcal{U}$ is the uniform distribution) which is B-rewind receiver private, then the OT protocol $\mathsf{OT}_{\mathsf{rep}}^{\mathsf{ListRew}}$ described in Figure 5.1 is a three-round sender list simulatable OT against a non-aborting receiver for the functionality $\mathcal{F}_{\mathsf{OT}}^{\mathcal{D}}$ (accordingly to Definition 3.2) otherwise, $\mathsf{OT}^{\mathsf{ListRew}}$ is sender private (accordingly to Definition 2.6). Moreover is $\mathsf{OT}_{\mathsf{rep}}^{\mathsf{ListRew}}$ a B-rewind receiver private OT protocol (accordingly to Definition 3.5). $\mathsf{OT}_{\mathsf{rep}}^{\mathsf{ListRew}}$ makes black-box use of $\mathsf{OT}^{\mathsf{ListRew}}$.*

Proof (sketch). The proof of this construction works almost analogous to the proof of $\mathsf{OT}^{\mathsf{ListRew}}$. In the reductions between the different hybrids, the challengers are simply replaced with the challengers corresponding to the parallel repeated version of the primitive. Since all the primitives used in this construction are parallel composable these challengers can easily be realized. We present the parallel adapted simulator in Figure 5.4.

The only parts in which the proof differs are the extraction probability in hybrid H_1 as well as the analysis of the simulator regarding the probability that the simulator executes more than maxrew rewinds in step 3 ("Extracting the input of R^*").

Sim^1 adopts for extractions the procedure of the underlying simulator $\mathsf{Sim}_i'^1$ for each i-th executions of $\mathsf{OT}^{\mathsf{ListRew}}$. Therefore, we are guaranteed that Sim^1 extracts the input of the receiver for the i-th executions of $\mathsf{OT}^{\mathsf{ListRew}}$ as soon as enough defenses are collected w.r.t. the first round $\mathsf{ot}_{i,1}$ (which are 4). We need now to bound the probability that after maxrew rewinds Sim^1 obtained enough defenses to apply $\mathsf{Sim}_i'^1$ for each i-th executions of $\mathsf{OT}^{\mathsf{ListRew}}$. Indeed, for these cases, we observe that instead of taking the probability of the simulator not being able to extract over all the m indexes and obtaining $1-\left(\frac{1}{4^m}\right)^{\mathsf{maxrew}}$ as the resulting probability, we need to further take the probability over all n executions and therefore we obtain $1-\left(\frac{1}{4^m}\right)^{\mathsf{maxrew}\cdot n}$ as the final success probability which is still overwhelming and therefore concludes the proof of the theorem. □

Figure 5.4: The simulator $\mathsf{Sim} = (\mathsf{Sim}^1, \mathsf{Sim}^2)$ for $\mathsf{OT}^{\mathsf{ListRew}}_{\mathsf{rep}}$.

Let $(\mathsf{Sim}_k'^1, \mathsf{Sim}_k'^2)$ be the simulator defined in Figure 5.2 for the k'th execution of $\mathsf{OT}^{\mathsf{ListRew}}$, which is parametrized by the distribution $\mathcal{D}$.
Input. The simulator Sim^1 takes as input the distribution $\mathcal{D}$ and 1^λ.

1) First thread: Sim^1 computes the following steps:
1. For all $i \in [m], k \in [n]$ and $b \in \{0,1\}$ upon receiving $\{\mathsf{ot}_{1,i}\}_{i\in[n]}$ from R^* generates $\mathsf{ot}_{2,i}$ as the honest sender would do. Send $\{\mathsf{ot}_{2,i}\}_{i\in[n]}$ to R^*.
2. If R^* does not send a 3rd round, abort and output the view of R^*.
3. Upon receiving $\{\mathsf{ot}_{3,i}\}_{i\in[n]}$ from R^*, initialize $\mathsf{Defense}_i$ with the set fo defenses of $\{\mathsf{ot}_{1,i}\}_{i\in[n]}$ obtained from $\{\mathsf{ot}_{3,i}\}_{i\in[n]}$ (the obtained defenses are defenses for the underlying oblivious transfer OT^{ListRew}).

2) Estimate the abort probability: Sim^1 initializes $\mathsf{ctr} := 0, T := 0$ and computes the following steps:
1. Increment T, i.e. $T := T+1$.
2. Compute and send a new 2nd round with fresh randomness to R^* as described in step (1).1 (i.e. as an honest sender would do).
3. If a valid 3rd round is received from R^* increment ctr, i.e. $\mathsf{ctr} := \mathsf{ctr}+1$.
4. If $\mathsf{ctr} = 12\lambda$ then output $p = \frac{12\lambda}{T}$, otherwise, perform a new rewind, i.e. go back to step (2).1.
5. Set $\mathsf{maxrew} = \lceil \frac{\lambda}{p} \rceil$.

3) Extracting the input of R^*: Sim^1 initializes $\mathsf{ctr} := 0$ and computes the following steps:
1. Perform the rewinds as follows:

Rewinding Threads: Sim computes the following steps:
(a) Increment ctr, i.e. $\mathsf{ctr} := \mathsf{ctr}+1$.
(b) Compute and send a new 2nd round with fresh randomness to R^* as described in step (1).1 (i.e. as an honest sender would do) and upon receiving $\{\widetilde{\mathsf{ot}}_{3,i}\}_{i\in[n]}$ from R^* check that the defenses $\widetilde{\mathsf{Defense}}_k$ contained in $\{\widetilde{\mathsf{ot}}_{3,i}\}_{i\in[n]}$ are valid w.r.t. $\{\mathsf{ot}_{1,i}\}_{i\in[n]}$ for all $i \in [n]$. Finally, set $\mathsf{Defense}_i := \mathsf{Defense}_i \cap \widetilde{\mathsf{Defense}}_i$.
(c) If $\mathsf{ctr} > \mathsf{maxrew}$ output fail.
(d) Check that $\mathsf{Sim}_i'^1$ is able to extract from $\mathsf{ot}_{1,i}$ using $\mathsf{Defense}_i$ for all $i \in [n]$. If this is the case, proceed to next step; else continue the rewinding, i.e. return to step (3).1.

2. Follow step (3).2 of $\mathsf{Sim}_i'^1$ and use the defenses contained in $\mathsf{Defense}_i$ to extract b^i.

4) Estimate the abort probability: Sim^1 initializes $\mathsf{ctr} := 0, T := 0$ and computes the following steps:

1. Increment T, i.e. $T := T+1$.
2. Compute a new 2nd round with fresh randomness for all $i \in [n]$, i.e. compute $\mathsf{ot}_{2,i}$ by executing the following steps:

   (a) Sample s_{b^i} at random from the distribution $\mathcal{D}$.
   (b) Generate $\mathsf{ot}_{2,i}$ as defined in steps (6.2) of $\mathsf{Sim}_i'^2$ w.r.t. input s_{b^i}.

   If a valid 3rd round is received from R^* increment ctr, i.e. $\mathsf{ctr} := \mathsf{ctr}+1$.
3. If $\mathsf{ctr} = 12\lambda$ then output $q = \frac{12\lambda}{T}$, otherwise, perform a new rewind, i.e. return to step (4).1.
4. Set $\mathsf{maxrew}' = \lceil \frac{\lambda}{q} \rceil$.

5) Query the ideal functionality: Sim^1 sends to ideal functionality $(\{b^i\}_{i\in[n]}, 1^d)$ receiving $(\vec{s}^1, \ldots, \vec{s}^d)$, where $\vec{s}^j = \{s^{b^i}\}_{i\in[n]}$ for $j \in [d]$, with $d := \lambda \cdot \mathsf{maxrew}'$.

6) Forcing the output Sim^2 on input $(\vec{s}^1, \ldots, \vec{s}^d)$ initializes $\mathsf{ctr} = 0$ and computes the following steps:

Threads to force the output:
1. Increment ctr, i.e. $\mathsf{ctr} := \mathsf{ctr}+1$
2. Compute $\{\mathsf{ot}_{2,i}\}_{i\in[n]}$ as described in step 4 "Estimate the abort probability" but using $\vec{s}^{\mathsf{ctr}}$ as input.
3. If $\mathsf{ctr} > \lambda \cdot \mathsf{maxrew}'$ output fail.
4. If R^* outputs an accepting third round, then output the view of R^* in this thread; else continue the rewinding, i.e. return to step (6).1.

Sim also keeps a count of its overall running time and if it reaches 2^λ steps it outputs fail.

5.4 Enhancing B-rewind Security of $\mathsf{OT}^{\mathsf{ListRew}}$

In Section 5 we described a 1-out-of-two OT $\mathsf{OT}^{\mathsf{ListRew}}$ which is sender list simulatable and B-rewind receiver private, where $B = 2$. The construction is described in Figure 5.1, and it uses as a building block a 1-out-of-2 OT OT' which is a sender list simulatable and only the receiver is private.

We note that it is possible to construct a 1-out-of-2 OT $\mathsf{OT}^{\mathsf{ListRew}}_{B'}$ which is sender list simulatable and B'-rewind receiver private, for a constant $B' = 6$ using the construction described in Figure 5.1, where the underlying OT used as a building block is already B-rewind receiver private. In more detail, the sender and receiver of $\mathsf{OT}^{\mathsf{ListRew}}_{B'}$ acts exactly as described in Figure 5.1 but using the algorithms of $\mathsf{OT}^{\mathsf{ListRew}}$ instead of OT'.

This observation leads to the following theorem.

Theorem 5.11. *Let X be a high min-entropy random variable defined by a probability distribution $\mathcal{D}$ and let $\mathsf{OT}^{\mathsf{ListRew}}$ the 1-out-of-two OT described in Figure 5.1 for the functionality $\mathcal{F}^{\mathcal{U}}_{\mathsf{OT}}$ (which is 3-rewind receiver private), then the OT protocol $\mathsf{OT}^{\mathsf{ListRew}}_{B'}$ described above is a three-round sender list simulatable OT against a sometimes aborting receivers for the functionality $\mathcal{F}^{\mathcal{D}}_{\mathsf{OT}}$ (according to Definition 3.2) otherwise, $\mathsf{OT}^{\mathsf{ListRew}}$ is sender private (according to Definition 2.6). Moreover $\mathsf{OT}^{\mathsf{ListRew}}_{B'}$ is a B'-rewind receiver private OT protocol (according to Definition 3.5), with $B' = 6$ and $\mathsf{OT}^{\mathsf{ListRew}}_{B'}$ makes black-box use of $\mathsf{OT}^{\mathsf{ListRew}}$.*

The proof follows similar to the one proven for Theorem 5.1.

5.5 Parallel Repetition

Besides being B-rewind secure, the presented OT protocol is also parallel composable. The parallel composability is important to construct the k-out-of-n OT protocol. The analysis of the parallel repetition is given in detail in Section 5.3 and the k-out-of-n OT protocol is presented in Section 6.

6 List Sender Simulatable and B-rewind Receiver Private k-out-of-n OT

In this section, we present a black-box compiler that turns a 1-out-of-2 oblivious transfer $\mathsf{OT}^{\mathsf{ListRew}}$ protocol into a k-out-of-n oblivious transfer $\mathsf{OT}^{\mathsf{ListRew}}_{k,n}$ protocol using a symmetric encryption scheme SE, while preserving the round complexity of the underlying OT protocol and maintaining the same security guarantees. Similar techniques are used in [SSR08], however, our setting is slightly different since we want a round-preserving transformation and stronger security guarantees. Namely, having as input $\mathsf{OT}^{\mathsf{ListRew}}$ that is sender list simulatable and receiver private we realize $\mathsf{OT}^{\mathsf{ListRew}}_{k,n}$ that has the same security guarantees, moreover, it is B-rewind secure against malicious senders. Our construction works in the simultaneous message model and is delayed input. That is, the receiver needs the input only to compute the second round and the sender requires the input to compute the last round. We also argue that if the delayed-input property is not needed, then we can obtain a k-out-of-n OT with the same security guarantees in the alternate message model.

Formally, we require the following tools:

– The three round 1-out-of-2 OT protocol $\mathsf{OT}^{\mathsf{ListRew}} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3, \mathsf{OT}_4 = (\mathsf{OT}^S_4, \mathsf{OT}^R_4))$ described in Figure 6.1.
– A k-out-of-n secret sharing scheme $\mathsf{TSS} = (\mathsf{Share}, \mathsf{Rec})$.
– A symmetric encryption scheme $\mathsf{SE} = (\mathsf{Enc}, \mathsf{Dec})$.

We describe the k-out-of-n oblivious transfer $\mathsf{OT}^{\mathsf{ListRew}}_{k,n}$ protocol in Figure 6.1.

Figure 6.1: Three round sender list simulatable and B-rewind receiver private k-out-of-n OT.

Initialization: The sender has input $(s_1, \ldots, s_n)$ and the receiver has in input a set $\mathcal{K}$ of k indices in $[n]$.

```
Round 1 (Receiver).
    1. Generate $\mathsf{ot}_{1,i} \coloneqq \mathsf{OT}_1(1^\lambda, b'_i)$ for all $i \in [n]$ with $b'_i \leftarrow \{0,1\}$.
    2. Send $\{\mathsf{ot}_{1,i}\}_{i\in[n]}$ to $S$.
Round 2.
    Sender:
        1. Sample random keys $\mathsf{k}'_{i,0}, \mathsf{k}'_{i,1} \leftarrow \{0,1\}^\lambda$ for all $i \in [n]$.
        2. Generate $\mathsf{ot}_{2,i} \leftarrow \mathsf{OT}_2(\mathsf{ot}_{1,i}, (\mathsf{k}'_{i,0}, \mathsf{k}'_{i,1}))$ for all $i \in [n]$.
        3. Send $\{\mathsf{ot}_{2,i}\}_{i\in[n]}$ to $R$.
    Receiver:
        1. Compute a $n$-bit string $b_1, \dots, b_n$ where $b_i = 1$ if $i \in \mathcal{K}$
        2. Compute $d_i = b_i \oplus b'_i$.
        3. Send $\{d_i\}_{i\in[n]}$ to $S$.
Round 3.
    Sender:
        1. Sample a random key $\mathsf{k} \leftarrow \{0,1\}^\lambda$.
        2. Encrypt all the inputs using the key $\mathsf{k}$, i.e. $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{k}, s_i)$ for all $i \in [n]$ and generate a
           $k$-out-of-$n$ secret sharing of the key, i.e. $\mathsf{k}_1, \dots, \mathsf{k}_n \leftarrow \mathsf{Share}(\mathsf{k}, n-k, n)$.
        3. For all $i \in [n]$, compute $\mathsf{ct}'_{i,0}$ and $\mathsf{ct}'_{i,1}$, where $\mathsf{ct}'_{i,0} \coloneqq \mathsf{k}'_{i,0} \oplus \mathsf{k}_i$ and $\mathsf{ct}'_{i,1} \coloneqq \mathsf{k}'_{i,1} \oplus \mathsf{ct}_i$ if $d_i = 0$
           and $\mathsf{ct}'_{i,0} \coloneqq \mathsf{k}'_{i,1} \oplus \mathsf{k}_i$ and $\mathsf{ct}'_{i,1} \coloneqq \mathsf{k}'_{i,0} \oplus \mathsf{ct}_i$ if $d_i = 1$.
        4. Send $\{\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1}\}_{i\in[n]}$ to $R$.
    Receiver:
        1. Compute $\mathsf{ot}_{3,i} \coloneqq \mathsf{OT}_3(\mathsf{ot}_{1,i}, \mathsf{ot}_{2,i})$ for all $i \in [n]$.
        2. Send $\{\mathsf{ot}_{3,i}\}_{i\in[n]}$ to $S$.
Offline computation.
    Sender:
        1. If $\bot = \mathsf{OT}_4^S(\mathsf{ot}_{1,i}, \mathsf{ot}_{2,i}, \mathsf{ot}_{3,i})$ for any $i \in [n]$ output $\bot$, else output $(s_1, \dots, s_n)$.
    Receiver:
        1. If $\bot = \mathsf{OT}_4^R(\mathsf{ot}_{1,i}, \mathsf{ot}_{2,i})$ for any $i \in [n]$ output $\bot$.
        2. Else generate $\mathsf{k}''_i \coloneqq \mathsf{OT}_3(\mathsf{ot}_{1,i}, \mathsf{ot}_{2,1})$ for all $i \in [n]$ and compute $\mathsf{k}'_i \coloneqq \mathsf{k}''_i \oplus \mathsf{ct}'_{i,0}$ for all
           $i \in [n]$ where $b_i \oplus d_i = 0$ and $\mathsf{ct}'_i \coloneqq \mathsf{k}''_i \oplus \mathsf{ct}'_{i,1}$ for all $i \in [n]$ where $b_i \oplus d_i = 1$.
        3. Compute $\mathsf{k}' \coloneqq \mathsf{Rec}(\{\mathsf{k}'_i\}_{i\in[n]:b_i\oplus d_i=0})$ and $s'_i \coloneqq \mathsf{Dec}(\mathsf{k}', \mathsf{ct}'_i)$ for all $i \in [n]$ where $b_i \oplus d_i = 1$.
        4. Output $(s'_i)_{i\in[n]:b_i\oplus d_i=1}$.
```

Correctness. The correctness of the special k-out-of-n OT follows from the correctness of the underlying OT protocol, the correctness of the symmetric encryption scheme as well as the correctness of the k-out-of-n secret sharing scheme. From the correctness of the OT protocol it follows that $\mathsf{k}'_i \coloneqq \mathsf{k}'_{i,b_i}$ for all $i \in [n]$. From the three round computation of the sender it follows that $\mathsf{ct}'_{i,0} \coloneqq \mathsf{k}'_{i,d_i} \oplus \mathsf{k}_i$ and $\mathsf{ct}'_{i,1} \coloneqq \mathsf{k}_{1\oplus d_i} \oplus \mathsf{ct}_i$ for all $i \in [n]$. For all $(n-k)$, $i \in [n]$, where $b_i \oplus d_i = 0 \Leftrightarrow b_i = d_i$, it holds that $\mathsf{k}'_i \oplus \mathsf{ct}'_{i,0} = \mathsf{k}'_{i,b_i} \oplus \mathsf{k}'_{i,d_i} \oplus \mathsf{k}_i = \mathsf{k}'_{i,b_i} \oplus \mathsf{k}'_{i,b_i} \oplus \mathsf{k}_i = \mathsf{k}_i$ and with the correctness of the k-out-of-n secret sharing scheme, it holds that $\mathsf{k} \coloneqq \mathsf{Rec}(\{\mathsf{k}'_i\}_{i:b_i\oplus d_i=0})$. For the remaining k, $i \in [n]$, where $b_i \oplus d_i = 1 \Leftrightarrow b_i = 1 \oplus d_i$, it holds that $\mathsf{k}'_i \oplus \mathsf{ct}'_{i,1} = \mathsf{k}'_{b_i} \oplus \mathsf{k}_{1\oplus d_i} \oplus \mathsf{ct}_i = \mathsf{k}'_{1\oplus d_i} \oplus \mathsf{k}_{1\oplus d_i} \oplus \mathsf{ct}_i = \mathsf{ct}_i$. Taking into account the correctness of the symmetric encryption scheme it follows that $s_i \coloneqq \mathsf{Dec}(\mathsf{k}, \mathsf{ct}_i)$ for all $i \in [n]$ such that $b_i \oplus d_i = 1$, which concludes the analysis of the correctness.

Theorem 6.1. *Let X be a high min-entropy random variable defined by a probability distribution $\mathcal{D}$ and let $\mathsf{OT}^{\mathsf{ListRew}}$ the 1-out-of-two OT described in Figure 5.1 for the functionality $\mathcal{F}_{\mathsf{OT}}^{\mathcal{U}}$ (where $\mathcal{U}$ is the uniform distribution) which is B-rewind receiver private, then the OT protocol $\mathsf{OT}_{k,n}^{\mathsf{ListRew}}$ described in Figure 6.1 is a three-round sender list simulatable OT against a non-aborting receiver for the functionality $\mathcal{F}_{\mathsf{OT}}^{\mathcal{D}}$ (accordingly to Definition 3.2) otherwise, $\mathsf{OT}^{\mathsf{ListRew}}$ is sender private (accordingly to Definition 2.6). Moreover is $\mathsf{OT}_{k,n}^{\mathsf{ListRew}}$ a B-rewind receiver private OT protocol (accordingly to Definition 3.5). $\mathsf{OT}_{k,n}^{\mathsf{ListRew}}$ makes black-box use of $\mathsf{OT}^{\mathsf{ListRew}}$.*

We split this theorem into three lemmas, the first two focusing on the simulatability and privacy against a malicious receiver and the third focusing on the privacy of the receiver against a malicious sender. These lemmas are proven in the corresponding sections Sections 6.1 to 6.3

6.1 List Simulatability Against Malicious Receivers

Lemma 6.2. *Let X be a high min-entropy random variable defined by a probability distribution $\mathcal{D}$ and let $\mathsf{OT}^{\mathsf{ListRew}}$ the 1-out-of-two OT described in Figure 5.1 for the functionality $\mathcal{F}_{\mathsf{OT}}^{\mathcal{U}}$ (where $\mathcal{U}$ is the uniform distribution) which is B-rewind receiver private, then the OT protocol $\mathsf{OT}_{k,n}^{\mathsf{ListRew}}$ described in Figure 6.1 is a three-round sender list simulatable OT against a non-aborting receiver for the functionality $\mathcal{F}_{\mathsf{OT}}^{\mathcal{D}}$ (accordingly to Definition 3.2).*

Proof. We prove that the OT $\mathsf{OT}_{k,n}^{\mathsf{ListRew}}$ described in Figure 6.2 is sender list simulatable by showing that a malicious (non-aborting) receiver is not able to distinguish between a real execution of the protocol and an ideal execution. The proof that this simulator runs in expected polynomial time and succeeds with overwhelming probability can be found in Claim 6.3 & Claim 6.4.

We prove the indistinguishability between the real world and the ideal world using a sequence of hybrids that we describe below:

Hybrid H_0: This hybrid corresponds to the real world.

Hybrid H_1: This hybrid is almost identical to the previous hybrid with the only difference that the input bit b of the receiver is extracted. In order to do so the hybrid follows the steps (1), (2) and (3) of Sim described in Figure 6.2, the hybrid outputs the view of R^* in the first thread. The indistinguishability of the hybrids follows with similar arguments to the one described in the claims 6.4, 6.3.

Hybrid H_1': This hybrid proceeds as in the previous hybrid except that after the extraction of the input the hybrid proceeds as explained in steps (4), and (6) of Sim described in Figure 6.2, but also in steps (4) and (6) the second round is computed honestly (as in steps (2) and (3)). The hybrid outputs the same output of Sim. The indistinguishability of the hybrids follows with similar arguments to the one described in the claims 6.4, 6.3.

Hybrid H_2^x: this hybrid, is described as in the previous hybrid but in step (x) the messages $\{\mathsf{ot}_{2,i}\}_{i\in}$ is generated as described in step (4).2 of Figure 6.2 (to be more precise, x is used to denote the 4th or the 6th stage of the simulation, namely $x = 4$ or $x = 6$). In other word $\{\mathsf{ot}_{2,i}\}_{i\in}$ is generated using the simulator $\overline{\mathsf{Sim}}^2$ w.r.t. input k'_{i,b'_i}). The indistinguishability of the hybrids follows from the sender's privacy the underlying OT protocol $\mathsf{OT}^{\mathsf{ListRew}}$, which follows with similar arguments to the one proven in Claim 6.5.

Hybrid H_3^x: this hybrid, is described as in the previous hybrid with the following differences.
The hybrid sets $\mathcal{K}$ as specified in step (3).2 of Figure 6.2, then for $i \in [n]$, the hybrid generates in step (x) the ciphertexts $\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1}$ as follows (to be more precise, x is used to denote the 4th or the 6th stage of the simulation, namely $x = 4$ or $x = 6$):

1. If $i \in \mathcal{K}$ set $\mathsf{ct}'_{i,0} \leftarrow \{0,1\}^\lambda$, while $\mathsf{ct}'_{i,1}$ is generated as in the previous hybrid.
2. If $i \notin \mathcal{K}$ set $\mathsf{ct}'_{i,1} \leftarrow \{0,1\}^\lambda$, while $\mathsf{ct}'_{i,0}$ is generated as in the previous hybrid.

If $|\mathcal{K}| \leq k$ then hybrid already corresponds to the ideal world, since all the information about the inputs s_i where $i \in [n]$ such that $i \notin \mathcal{K}$ is removed from the protocol execution. Otherwise, we proceed with the following hybrids.
The indistinguishability between this and the previous hybrid follows from the perfect security of (multiple instances of) the one-time pad, a similar reduction is shown in Claim 6.6.

In both of the following hybrids, it is assumed that $|\mathcal{K}| > k$.

Hyrid H_4^x: this hybrid, is described as in the previous hybrid but the hybrid in step (x) switches from the generation of a $(n-k)$-out-of-n secret sharing of the symmetric encryption key k to the generation of a $(n-k)$-out-of-n secret sharing of a random value (to be more precise, x is used to denote the 4th or the 6th stage of the simulation, namely $x = 4$ or $x = 6$). The indistinguishability between this and the previous hybrid follows from the security of the k-out-of-n secret sharing scheme. A similar reduction is shown in Claim 6.7.

Hybrid H_5^x: this hybrid, is described as in the previous hybrid but the hybrid in step (x) switches from symmetric encryptions of the actual inputs s_i's, i.e. $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{k}, s_i)$ to encryptions of random values, i.e. $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{k}, r_i), r_i \leftarrow \{0,1\}^\lambda$ (to be more precise, x is used to denote the 4th or the 6th stage of the simulation, namely $x = 4$ or $x = 6$). The indistinguishability between this and the previous hybrid follows from the IND-CPA security of the underlying symmetric encryption scheme. For this subcase, this hybrid corresponds to the ideal world. A similar reduction is shown in Claim 6.8.

From the above arguments we can conclude that $\mathsf{H}_0 \approx \mathsf{H}_1 \approx \mathsf{H}_1' \approx_c \mathsf{H}_2^4 \approx \mathsf{H}_3^4 \approx \mathsf{H}_4^4 \approx_c \mathsf{H}_5^4 \approx_c \mathsf{H}_2^6 \approx \mathsf{H}_3^6 \approx \mathsf{H}_4^6 \approx_c \mathsf{H}_5^6$.

□

Figure 6.2: Simulator $\mathsf{Sim} = (\mathsf{Sim}^1, \mathsf{Sim}^2)$ for $\mathsf{OT}_{k,n}^{\mathsf{ListRew}}$

Let $\overline{\mathsf{Sim}} = (\overline{\mathsf{Sim}}^1, \overline{\mathsf{Sim}}^2)$ be the simulator defined in Figure 5.4 parametrized by the uniform distribution.
Input. The simulator Sim^1 takes as input the distribution $\mathcal{D}$ and 1^λ.

1) First thread Sim^1 computes the following steps:
1. Upon receiving $\{\mathsf{ot}_{1,i}\}_{i\in[n]}$ from R^* compute $\{\mathsf{ot}_{2,i}\}_{i\in[n]}$ as the honest sender would do and send them to R^*.
2. Upon receiving $\{d_i\}_{i\in[n]}$ from R^* compute $\{\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1}\}_{i\in[n]}$ as the honest sender would do and send them to R^*.
3. If R^* does not send any 3rd round then abort giving in output the view of R^*.
4. Upon receiving $\{\mathsf{ot}_{3,i}\}_{i\in[n]}$ from R^* initialize $\mathsf{Defense}$ with the set of defenses of $\{\mathsf{ot}_{1,i}\}_{i\in[n]}$ obtained from $\{\mathsf{ot}_{3,i}\}_{i\in[n]}$.

2) Estimate abort probability Sim initializes $\mathsf{ctr} = 0, T = 0$ and computes the following steps:
1. Increment $T = T + 1$.
2. Compute and send to R^* a new 2nd round with fresh randomness as an honest sender would do. Upon receiving a 2nd round from R^*, compute and send to R^* a new 2nd round with fresh randomness as an honest sender would do. If a valid 3rd round is received from R^* increment $\mathsf{ctr} = \mathsf{ctr} + 1$.
3. If $\mathsf{ctr} = 12\lambda$ then output $p = \frac{12\lambda}{T}$, otherwise, perform a new rewind, i.e. go back to step (3).1.
4. Set $\mathsf{maxrew} = \lceil \frac{\lambda}{p} \rceil$.

3) Extracting input of R^* Sim initializes $\mathsf{ctr} = 0$ and computes the following steps:
1. Perform the rewinds as follows:

   Rewinding Threads Sim computes the following steps:
   (a) Increment $\mathsf{ctr} = \mathsf{ctr} + 1$
   (b) Repeat steps of Rounds 2 and Round 3 with fresh randomness and upon receiving $\{\widetilde{\mathsf{ot}_{3,i}}\}_{i\in[m]}$ from R^* check that the defenses contained in $\{\widetilde{\mathsf{ot}_{3,i}}\}_{i\in[m]}$ are valid w.r.t. $\{\mathsf{ot}_{1,i}\}_{i\in[n]}$. Let $\widetilde{\mathsf{Defense}}$ be the set of valid defenses in $\{\widetilde{\mathsf{ot}_{3,i}}\}_{i\in[m]}$. Finally, set $\mathsf{Defense} := \mathsf{Defense} \cap \widetilde{\mathsf{Defense}}$.
   (c) If $\mathsf{ctr} > \mathsf{maxrew}$ output fail.
   (d) Check if it is possible to extract $b'_1, \ldots, b'_n$ played by R^* in $\{\mathsf{ot}_{1,i}\}_{i\in[n]}$ following the same steps of the simulator $\overline{\mathsf{Sim}}^1$ described in Figure 5.4.; else continue the rewinding, i.e. go to step (2).1.

2. Follow the same steps of the simulator $\overline{\mathsf{Sim}}^1$ described in Figure 5.4.on input the defenses $\mathsf{Defense}$ to extract the bit string $b'_1, \ldots, b'_n$ played by R^* in $\{\mathsf{ot}_{1,i}\}_{i\in[n]}$. Then use $d_1, \ldots, d_n$ (sent by R^* in the 2nd round of "First Thread") to compute the input $b_1, \ldots, b_n$ of R^*. Finally, compute the set $\mathcal{K}$ as the indices i where $b'_i \oplus d_i = 1$, for $i \in [n]$.

4) Estimate abort probability Sim initializes $\mathsf{ctr} = 0, T = 0$ and computes the following steps:
1. Increment $T = T + 1$.
2. For $i \in [n]$ generate $\mathsf{ot}_{2,i}$ as defined in step 4 of $\overline{\mathsf{Sim}}^2$ w.r.t. input k'_{i,b'_i}, where k'_{i,b'_i} is taken at random from the uniform distribution. Send $\{\mathsf{ot}_{2,i}\}_{i\in[n]}$ to R^*.
3. Upon receiving the 2nd round from R^* compute $(\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})_{i\in[n]}$ as follows:
   (a) If $|\mathcal{K}| > k$
      i. Sample a random symmetric encryption key k and compute $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{k}, r_i)$ for all $i \in [n]$ with $r_i \leftarrow \{0,1\}^\lambda$.
      ii. Generate $\mathsf{k}_1, \ldots, \mathsf{k}_n$ as an n-out-of-n secret sharing of a random $r \leftarrow \{0,1\}^\lambda$.
   (b) If $|\mathcal{K}| \le k$
      i. Sample $\{s_i\}_{i\in\mathcal{K}}$ at random from distribution $\mathcal{D}$.
      ii. Sample a random symmetric encryption key k and compute $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{k}, s_i)$ for all $i \in \mathcal{K}$.
      iii. Generate $\mathsf{k}_1, \ldots, \mathsf{k}_n$ as an n-out-of-n secret sharing of a k.
   (c) For all $i \in \mathcal{K}$ sample $\mathsf{ct}'_{i,0} \leftarrow \{0,1\}^\lambda$ and generate $\mathsf{ct}'_{i,1}$ as described in the protocol using ct_i.
   (d) For all $i \notin \mathcal{K}$ sample $\mathsf{ct}'_{i,1} \leftarrow \{0,1\}^\lambda$ and generate $\mathsf{ct}'_{i,0}$ as described in the protocol using k_i.

Send $(\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})_{i\in[n]}$ to R^* and if a valid 3rd round is received from R^* increment $\mathsf{ctr} = \mathsf{ctr} + 1$.
4. If $\mathsf{ctr} = 12\lambda$ then output $q = \frac{12\lambda}{T}$, otherwise, perform a new rewind, i.e. go back to step (4).1.
5. Set $\mathsf{maxrew}' = \lceil \frac{\lambda}{q} \rceil$.

5) Query the ideal functionality If $|\mathcal{K}| \leq k$ Sim sets $d = \lambda \cdot \mathsf{maxrew}'$ and sends to the ideal functionality $(\mathcal{K}, 1^d)$ obtaining $(\vec{s}^1, \ldots, \vec{s}^d)$, where $\vec{s}^j = \{s_i\}_{i\in\mathcal{K}}$ for $j \in [d]$.

6) Forcing the output Sim^2 on input $(\vec{s}^1, \ldots, \vec{s}^d)$ computes the following steps:

Threads for forcing the output Sim^2 initializes $\mathsf{ctr} = 0$ and computes the following steps
1. Increment $\mathsf{ctr} = \mathsf{ctr} + 1$
2. Compute $(\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})_{i\in[n]}$ as describe in Step 4 "Estimate abort probability" but using $\vec{s}^{\mathsf{ctr}}$ as input for the case where $|\mathcal{K}| \leq k$.
3. If $\mathsf{ctr} > \lambda \cdot \mathsf{maxrew}'$ output fail.
4. If R^* gives an accepting Round 3 output the view of R^* in this thread; else go to step (6).1.

Sim also keeps a count of its overall running time and if it reaches 2^λ steps it output fail.

Claim 6.3 Sim *runs in expected polynomial time in* λ.

The analysis of the simulator follows similarly to the analysis in [GK96] as recapped in [Lin16]. We analyze the running time in more detail. Let ε be the probability that the adversary R^* outputs a valid third round. The following analysis takes into account that each time Sim follows the steps of $\overline{\mathsf{Sim}}$ it runs in polynomial time.

In the first step ("First thread") the simulator clearly runs in polynomial time. The estimation of the probability for the second step ("Estimate the abort probability") is executed until Sim does not collect 12λ valid third rounds from R^*. Since the probability that R^* outputs, a valid third round is ε, Sim stops after $12\lambda/\varepsilon$ attempts.

From the analysis in [GK96] it follows that the estimation p is within a constant factor from ε, unless of a negligible probability $2^{-\lambda}$ (therefore the case of running 2^λ steps adds only a polynomial amount of overhead to the simulator).

This results in an overall running time of the simulator that is expected polynomial in λ.

Claim 6.4 Sim *outputs* fail *with negligible probability in* λ.

There are three possibilities that the simulator Sim outputs fail:

1. The simulator executes more than 2^λ steps
2. The simulator executes more than maxrew rewinds in step 3 ("Extracting the input of R^*")
3. The simulator executes more than $\lambda \cdot \mathsf{maxrew}'$ rewinds in step 6 ("Forcing the output")

The first case is directly bounded by the previous proof, i.e. if the simulator runs in expected polynomial time in λ then it only executes more than 2^λ steps with negligible probability.

To show that the second case is negligible, we need to bound the probability with which Sim extracts the input of R^*. The extraction of the input from R^* is performed by Sim^1 following the extraction procedure of $\overline{\mathsf{Sim}}^1$. Therefore if Sim^1 fails the extraction with non-negligible probability, it is possible to show a reduction against the parallel (simulation) security of the underlying $\mathsf{OT}^{\mathsf{ListRew}}$.

To bound the probability of the third case we first prove that the probability with which R^* is answering after receiving a simulated 2nd and 3rd rounds (as described in step (4)) is negligible close to the probability that R^* is answering after receiving a 2nd and 3rd round of the protocol computed honestly.

In order to do so we consider the following hybrids:

Hybrid H_0: This hybrid corresponds to the real world execution.

Hybrid H'_0: The hybrids act exactly as the previous hybrid but upon receiving receiver's messages $\mathsf{ot}_{1,1}, \ldots, \mathsf{ot}_{1,n}$ the input bit b'_i of the receiver is extracted in exponential time. Then the hybrid sends the 2nd round of the sender (computed in an honest way) to R^* and upon receiving $d_1, \ldots, d_n$ round from R^* the set $\mathcal{K}$ is computed as the indices i where $b'_i \oplus d_i = 1$, for $i \in [n]$. The rest of the protocol is executed as the honest sender would do.

Hybrid H_1: this hybrid is described as in the previous hybrid but the messages $\{\mathsf{ot}_{2,i}\}_{i\in}$ is generated as described in step (4).2 of Figure 6.2 (i.e. using the simulator $\overline{\mathsf{Sim}}^2$ w.r.t. input k'_{i,b'_i}). The indistinguishability of the hybrids follows from the sender's privacy the underlying OT protocol $\mathsf{OT}^{\mathsf{ListRew}}$, which we formally prove in Claim 6.5.

Hybrid H_2: this hybrid, is described as in the previous hybrid with the following differences. The hybrid sets $\mathcal{K}$ as specified in step (3).2 of Figure 6.2, then for $i \in [n]$, the hybrid generates the ciphertexts $\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1}$ as follows:

1. If $i \in \mathcal{K}$ set $\mathsf{ct}'_{i,0} \leftarrow \{0,1\}^\lambda$, while $\mathsf{ct}'_{i,1}$ is generated as in the previous hybrid.
2. If $i \notin \mathcal{K}$ set $\mathsf{ct}'_{i,1} \leftarrow \{0,1\}^\lambda$, while $\mathsf{ct}'_{i,0}$ is generated as in the previous hybrid.

If $|\mathcal{K}| \leq k$ then hybrid already corresponds to the ideal world, since all the information about the inputs s_i where $i \in [n]$ such that $i \notin \mathcal{K}$ is removed from the protocol execution. Otherwise, we proceed with the following hybrids.

The indistinguishability between this and the previous hybrid follows from the perfect security of (multiple instances of) the one-time pad. The reduction is shown in Claim 6.6.

In both of the following hybrids, it is assumed that $|\mathcal{K}| > k$.

Hyrid H_3: this hybrid, is described as in the previous hybrid but the hybrid switches from the generation of a $(n-k)$-out-of-n secret sharing of the symmetric encryption key k to the generation of a $(n-k)$-out-of-n secret sharing of a random value. The indistinguishability between this and the previous hybrid follows from the security of the k-out-of-n secret sharing scheme. The reduction is shown in Claim 6.7.

Hybrid H_4: this hybrid, is described as in the previous hybrid but the hybrid switches from symmetric encryptions of the actual inputs s_i's, i.e. $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{k}, s_i)$ to encryptions of random values, i.e. $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{k}, r_i), r_i \leftarrow \{0,1\}^\lambda$. The indistinguishability between this and the previous hybrid follows from the IND-CPA security of the underlying symmetric encryption scheme. For this subcase, this hybrid corresponds to the ideal world. The reduction is shown in Claim 6.8.

Claim 6.5 *Let the parallel version of $\mathsf{OT}^{\mathsf{ListRew}}$ be a three-round seder private OT protocol (against aborting malicious receiver), then the hybrids H'_0 and H_1 are computationally indistinguishable.*

Proof. To prove that the hybrids are computationally indistinguishable, we rely on the simulator of the underlying OT protocol $\mathsf{OT}^{\mathsf{ListRew}}$. We prove this claim by contradiction. Let CH be the challenger of the sender private $\mathsf{OT}^{\mathsf{ListRew}}$. We define the adversary $\mathcal{A}$ for $\mathsf{OT}^{\mathsf{ListRew}}$ which internally runs R^* and acts as a proxy between R^* and the challenger with respect to the messages of one execution of $\mathsf{OT}^{\mathsf{ListRew}}$. In more detail, $\mathcal{A}$ upon receiving $\{\mathsf{ot}_{1,i}\}_{i\in[n]}$ from R^* sends them to the challenger along with $\{\mathsf{k}'_{i,0}, \mathsf{k}'_{i,1}\}_{i\in[n]}$, where are chosen at random. $\mathcal{A}$ upon receiving $\{\mathsf{ot}_{2,i}\}_{i\in[n]}$ from the challenger sends them to R^* and continues the rest of the protocol as an honest sender would do, using as keys $\{\mathsf{k}'_{i,0}, \mathsf{k}'_{i,1}\}_{i\in[n]}$ in the third round. The reduction runs the distinguisher (which exists by contradiction) on the output of $\mathcal{A}$. The reduction gives as output the output of the distinguisher.

□

Claim 6.6 *The hybrids H_1 and H_2 are perfectly indistinguishable.*

Proof. In this proof, we denote the number of times for which $b'_i \oplus d_i = 1$ as ℓ, i.e. $|\mathcal{K}| = \ell$. We prove this claim by contradiction. We assume that there exists an adversary R^*, that can distinguish between the two hybrids H_1 and H'_2, with non-negligible probability, then we can use R^* to construct an adversary $\mathcal{A}$ that breaks the underlying $(n-\ell)$ or ℓ instances of the one-time pads.

Upon receiving $\{\mathsf{ot}_{1,i}\}_{i\in[n]}$ from R^* the adversary $\mathcal{A}$ runs in exponential time and extracts bits b'_i for all $i \in [n]$. For the generation of the second round, namely of $\{\mathsf{ot}_{2,i}\}_{i\in[n]}$, the reduction behaves as specified in the previous hybrid. Observe that after this step, there exists no information about the keys $\mathsf{k}'_{1\oplus b'_i}$ for all $i \in [n]$ in the OT execution. In the next step, for the generation of the third round of the sender, $\mathcal{A}$ behaves as described in the protocol until the generation of the ciphertexts.

Now, we distinguish between two cases:

1. $b_i = b'_i \oplus d_i = 1$ for less than or equal to k i's, i.e. $|\mathcal{K}| \leq k$.
2. $b_i = b'_i \oplus d_i = 1$ for more than k i's, i.e. $|\mathcal{K}| > k$.

In the first case, the generation of the ciphertexts $\mathsf{ct}'_{i,0}$ for al $i \in [n]$ happens as described in the protocol execution. For the generation of the ciphertexts $\mathsf{ct}'_{i,1}$ the adversary $\mathcal{A}$ interacts with a challenger that executes ℓ instances of a one-time pad.[14] In more detail, it generates $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{k}, s_i)$ and samples $r_i \leftarrow \{0,1\}^\lambda$ for all $i \in [n]$ such that $b'_i \oplus d_i = 0$. It then sends $(\mathsf{ct}_i, r_i)_{i\in[n]:b'_i\oplus d_i=0}$ as its $(n-\ell)$ challenges to the underlying challenger of the one-time pad and receives as a reply $\mathsf{ct}'_{i,1}$, which $\mathcal{A}$ uses as a reply of the third round for $\mathcal{A}'$. The remaining ciphertexts $\mathsf{ct}'_{i,1}$ are generated as described in the previous hybrid.

In this case, the reduction concludes with the observation that if the challenger of the $(n-\ell)$ one-time pad instances generates the ciphertexts $\mathsf{ct}'_{i,1}$ using ct_i for all $i \in [n]$ such that $b'_i \oplus d_i = 0$, then this corresponds to hybrid H_1 and if the challenger generates $\mathsf{ct}'_{i,1}$ using r_i for all $i \in [n]$ such that $b'_i \oplus d_i = 0$, then this corresponds to hybrid H_2. If the adversary $\mathcal{A}'$ is now able to distinguish between H_1 and H_2 with non-negligible probability, then our constructed adversary $\mathcal{A}$ breaks the underlying $(n-\ell)$ one-time pad instances with non-negligible probability. This results in a contradiction and concludes the proof for this case. In this case, Lemma 6.2 already follows since all the information about the inputs s_i where $i \in [n]$ such that $b'_i \oplus d_i = 0$ is removed from the protocol execution.

In the second case, the generation of the ciphertexts $\mathsf{ct}'_{i,1}$ for al $i \in [n]$ happens as described in the protocol execution. For the generation of the ciphertexts $\mathsf{ct}'_{i,0}$ the adversary $\mathcal{A}$ interacts with a challenger that executes ℓ instances of a one-time pad. In more detail, it samples $r_i \leftarrow \{0,1\}^\lambda$ for all $i \in [n]$ such that $b'_i \oplus d_i = 1$. It then sends $(\mathsf{k}_i, r_i)_{i\in[n]:b'_i\oplus d_i=1}$ as its ℓ challenges to the underlying challenger of the one-time pad and receives as a reply $\mathsf{ct}'_{i,0}$, which $\mathcal{A}$ uses as a reply of the third round for $\mathcal{A}'$. The remaining ciphertexts $\mathsf{ct}'_{i,0}$ are generated as described in the previous hybrid.

In this case, the reduction concludes with the observation that if the challenger of the ℓ one-time pad instances generates the ciphertexts $\mathsf{ct}'_{i,0}$ using k_i for all $i \in [n]$ such that $b'_i \oplus d_i = 1$, then this corresponds to hybrid H_1 and if the challenger generates $\mathsf{ct}'_{i,0}$ using r_i for all $i \in [n]$ such that $b'_i \oplus d_i = 1$, then this corresponds to hybrid H_2. If the adversary $\mathcal{A}'$ is now able to distinguish between H_1 and H_2 with non-negligible probability, then our constructed adversary $\mathcal{A}$ breaks the underlying ℓ one-time pad instances with non-negligible probability. This results in a contradiction and concludes the proof for this case. In this case, to conclude the proof of the lemma, we need to proceed using the following hybrids. □

Claim 6.7 *If $b'_i \oplus d_i = 1$ for more than k i's (i.e. $|\mathcal{K}| > k$), then the hybrids H_2 and H_3 are perfectly indistinguishable.*

Proof. In this proof, we assume that there exist exactly $(n-k-1)$ indices for which $b'_i \oplus d_i = 0$. We prove this claim by contradiction. We assume that there exists an adversary $\mathcal{A}'$, that can distinguish between the two hybrids H_2 and H_3, in the case that $b_i \oplus d_i = 1$ for more than k i's, with non-negligible probability, then we can use $\mathcal{A}'$ to construct an adversary $\mathcal{A}$ that breaks the underlying k-out-of-n secret sharing scheme.

Upon receiving $\{\mathsf{ot}_{1,i}\}_{i\in[n]}$ from R^* the adversary $\mathcal{A}$ runs in exponential time and extracts bits b'_i for all $i \in [n]$. For the generation of the second and third round the reduction behaves as specified in the previous hybrid until the generation of the shares for the symmetric encryption k. Instead of generating the shares by itself, $\mathcal{A}$ samples a random value $r \leftarrow \{0,1\}^\lambda$ and sends (k, r) to the underlying challenger of a $(n-k)$-out-of-n secret sharing scheme that replies with $\{\mathsf{k}_i\}_{i\in[n]:b'_i\oplus d_i=0}$. Due to the previous hybrid, all the remaining ciphertexts, i.e. $\{\mathsf{k}_i\}_{i\in[n]:b'_i\oplus d_i=1}$, already correspond to random values. For the generation of the remaining ciphertexts and for the remaining steps, $\mathcal{A}$ proceeds as described in the previous hybrid.

We conclude the reduction with the observation that if the challenger of the $(n-k)$-out-of-n secret sharing scheme generates the shares w.r.t k, then $\mathcal{A}$ simulates hybrid H_2 and if the challenger of the $(n-k)$-out-of-n secret sharing scheme generates the shares w.r.t the random value r, then this corresponds to hybrid H_3. If

[14] $(n-\ell)$ instances here means that the challenger flips a single bit b and then encrypts, in all $(n-\ell)$ executions of the one-time pad, the challenge message corresponding to the bit b. This directly reduces to the single instance case using a simple hybrid argument.

the adversary $\mathcal{A}'$ is now able to distinguish between H_2 and H_3 with non-negligible probability, then our constructed adversary $\mathcal{A}$ breaks the underlying $(n-k)$-out-of-n secret sharing scheme with non-negligible probability. This results in a contradiction and concludes the proof. □

Claim 6.8 *If $b'_i \oplus d_i = 1$ for more than k i's (i.e. $|\mathcal{K}| > k$), and* SE *is an IND-CPA secure symmetric encryption scheme, then the hybrids H_3 and H_4 are computationally indistinguishable.*

Proof. We prove this claim by contradiction. We assume that there exists an adversary R^*, that can distinguish between the two hybrids H_3 and H_4, in the case that $b_i \oplus d_i = 1$ for more than k i's, with non-negligible probability, then we can use R^* to construct an adversary $\mathcal{A}$ that breaks the underlying IND-CPA secure symmetric encryption scheme.

Let us fix the first round message $\{\mathsf{ot}_{1,i}\}_{i\in[n]}$ that maximizes the distinguishing advantage of the adversary R^* in the above two experiments. Note that in $\mathsf{OT}^{\mathsf{ListRew}}$ the input of the receiver is fixed in the first round, therefore it is fixed once $\{\mathsf{ot}_{1,i}\}_{i\in[n]}$ are generated. Let b'_i be the inputs of the receiver, respectively, used in $\{\mathsf{ot}_{1,i}\}_{i\in[n]}$. We define the adversary $\mathcal{A}$ for OT' which internally runs R^* and acts as a proxy between R^* and the challenger of the IND-CPA security.

In more details, $\mathcal{A}$ takes as auxiliary input $(\{\mathsf{ot}_{1,i}\}_{i\in[n]}, \{b'_i\}_{i\in[n]})$ and compute the following steps. $\mathcal{A}$ generate the 2nd round as described in the previous hybrid, then in the third round $\mathcal{A}$ samples a $(n-k)$-out-of-n secret sharing of a random value and uses those values in the generation of the ciphertexts $\mathsf{ct}'_{i,0}$. To obtain the ciphertexts ct_i the adversary submits as challenge queries to its underlying challengers the pairs (s_i, r_i) for all $i \in [n]$, with $r_i \leftarrow \{0,1\}^\lambda$ to obtain as a reply ct_i for all $i \in [n]$. It then uses these ciphertexts as in the protocol description for the generation of the ciphertexts $\mathsf{ct}'_{i,1}$. Finally, $\mathcal{A}$ outputs $(\mathsf{ct}'_{i,0}, \mathsf{ct}'_{i,1})$ to $\mathcal{A}'$.

We conclude the reduction with the observation that if the challenger of the IND-CPA secure encryption generates all the ciphertexts ct_i w.r.t the sender inputs $\{s_i\}_{i\in[n]}$, then $\mathcal{A}$ simulates hybrid H_3 and if the challenger of the symmetric encryption scheme generates the ciphertexts w.r.t. the random values r_i, then this corresponds to hybrid H_4. If the adversary R^* is now able to distinguish between H_3 and H_4 with non-negligible probability, then our constructed adversary $\mathcal{A}$ breaks the IND-CPA security of the underlying symmetric encryption scheme with non-negligible probability. This results in a contradiction and concludes the proof of the claim. Since this hybrid corresponds to the ideal world, this concludes also the proof of Lemma 6.2. □

6.2 Proof of the Sender Privacy

Theorem 6.9. *Let X be a high min-entropy random variable defined by a probability distribution $\mathcal{D}$ and let $\mathsf{OT}^{\mathsf{ListRew}}$ the 1-out-of-two OT described in Figure 5.1 then the OT protocol $\mathsf{OT}^{\mathsf{ListRew}}_{k,n}$ described in Figure 6.1 is sender private (accordingly to Definition 2.6) against aborting adversary.*

Proof (Sketch). This lemma can be proven in the same way as Lemma 5.8. Therefore, we refer to Lemma 5.8 for further details. □

6.3 Proof of B-rewind Receiver Security

Lemma 6.10. $\mathsf{OT}^{\mathsf{ListRew}}$ *the 1-out-of-two OT described in Figure 5.1 which is B-rewind receiver private, then $\mathsf{OT}^{\mathsf{ListRew}}_{k,n}$ is B-rewind receiver private OT protocol (accordingly to Definition 3.5).*

Proof (Sketch). This lemma can be proven similarly to the bullet (3) of Claim 6.4. □

Remark 6.11. We finally observe that if the delayed-input property is not needed, then the k-out-of-n OT described in Figure 6.1 can be easily modified to work in the alternated model. Indeed, if the receiver knows the input in the first round already, then it knows the shares $\{d_i\}_{i\in[n]}$, and these can be sent in the clear already in the first round (instead of waiting until the second round). Accordingly, the sender that receives these shares can compute all its messages already in the second round. Hence, the only party speaking in the third round will be the receiver.

7 Two-Party Computation

In this section, we construct a three-round (in the simultaneous message model) 2-party computation protocol Π^{2PC} that is 1-rewinding secure list-simulatable with delayed function selection for NC_1 circuits (we refer to Definition 3.3 for a formal definition).

To construct our protocol, we make use of the following tools.

– A two-round information-theoretic two client, m server MPC protocol $\Phi = (\Phi_1, \{\Phi_{2,i}\}_{i\in[m]})$[15] for computing NC^1 circuits, that satisfies security with selective abort against a malicious, adaptive adversary corrupting a single client and at most $t = (m-1)/3$. Here, we fix $m = 8\lambda$. We need the protocol to satisfy two additional properties:
  - We require the first round message of the protocol to be independent of the function description and only depend on its size.
  - Given a first round message from the clients, the servers in the second round, are given descriptions of two functions f_0, f_1 and should be able to generate the second-round message corresponding to these two functions on the same first-round message from the client. In other words, we require the protocol's first round message to be reusable once.

  In [IKSS21] the authors argue how to obtain such a scheme.
– A PRF $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^*$.
– A symmetric encryption scheme $\mathsf{SE} = (\mathsf{Enc}, \mathsf{Dec})$.
– A delayed-input, 1-rewinding secure extractable commitment scheme $\mathsf{ExtCom} = (\mathsf{ExtCom}_1, \mathsf{ExtCom}_2, \mathsf{ExtCom}_3)$ which has been presented in Section 2.2.
– A three-round list simulatable against sometimes aborting adversaries λ-out-of-m OT protocol $\mathsf{OT}^{\lambda,m} := (\mathsf{OT}_1^{\lambda,m}, \mathsf{OT}_2^{\lambda,m}, \mathsf{OT}_3^{\lambda,m}, \mathsf{OT}_{\mathsf{out}}^{\lambda,m})$. We require the protocol to be $B^{\mathsf{ot}} = 6$ rewind secure against corrupted receivers for the uniform distribution $\mathcal{U}$, i.e. for the functionality $\mathcal{F}_{\mathsf{OT}}^{\mathcal{U}}$. In more detail, the uniform distribution $\mathcal{U}$ here samples m random strings of length λ each acting as the randomness s_i for all $i \in [m]$ used for the commitment generated in the first round of the protocol.
– A three-round list simulatable against sometimes aborting corrupted receivers, and private against corrupted sender 1-out-of-2 OT protocol $\mathsf{OT}^{1,2} := (\mathsf{OT}_1^{1,2}, \mathsf{OT}_2^{1,2}, \mathsf{OT}_3^{1,2}, \mathsf{OT}_{\mathsf{out}}^{1,2})$ that is $B^{\mathsf{ot}} = 6$ rewind secure against corrupted receiver for the uniform distribution $\mathcal{U}'$, i.e. for the functionality $\mathcal{F}_{\mathsf{OT}}^{\mathcal{U}'}$. In more detail, the uniform distribution $\mathcal{U}'$ here samples two random strings of length λ acting as the secret keys $(\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k})$ used for the ciphertexts generated in the third round of the protocol.
– A garbling scheme $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$.

We propose the formal construction of our protocol in Figure 7.1, and refer to the introduction for an informal description of the protocol.

Figure 7.1: The three rounds of the 2PC protocol Π^{2PC}

Initialization: The sender S has input y and the receiver R has input x.

Round 1.

Sender:

1. Compute $\mathsf{com}_1^i := \mathsf{ExtCom}_1(1^\lambda; s_i)$ with $s_i \leftarrow \{0,1\}^*$ for all $i \in [m]$.
2. Send $(\{\mathsf{com}_1^i\}_{i\in[m]})$ to R.

Receiver:

1. Compute $(x_1, \ldots, x_m) \leftarrow \Phi_1(x)$. We assume that $x_i \in \{0,1\}^\ell$ and we denote by $x_{i,j}$ the j-th bit of x_i fo r all $j \in [\ell]$.
2. Sample $x_{i,j,1}, \ldots, x_{i,j,\lambda-1} \leftarrow \{0,1\}$ and set $x_{i,j,\lambda-1} := x_{i,j} \oplus_{k\in[\lambda-1]} x_{i,j,k}$ for all $i \in [m], j \in [\ell]$.

[15] Here, $\Phi_{2,i}$ is the function that takes $\{y_{i,j}\}_{j\in[\ell]}$ and $\{x_{i,j,k}\}_{j\in[\ell],k\in[\lambda]}$ as inputs and first reconstructs $\{x_{i,j}\}_{j\in[\ell]}$ and then applies the function computed by the i-th server in the outer protocol Φ on $f, \{x_{i,j}, y_{i,j}\}_{j\in[\ell]}$

3. Choose a random subset $K \subset [m]$.
4. For all $i \in [m], j \in [\ell], k \in [\lambda]$ sample $r_{i,j,k} \leftarrow \{0,1\}^\lambda$ and compute $\mathsf{ot}_1^{i,j,k} \leftarrow \mathsf{OT}_1^{1,2}(1^\lambda, x_{i,j,k})$. Sample $r \leftarrow \{0,1\}^\lambda$ and compute $\mathsf{ot}_1 \leftarrow \mathsf{OT}_1^{\lambda,m}(1^\lambda, K; r)$.
5. Send ot_1^i and $\{\mathsf{ot}_1^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to S.

Round 2.

Sender:

1. Sample $\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k} \leftarrow \{0,1\}^\lambda$ for all $i \in [m], j \in [\ell]$ and $k \in [\lambda]$.
2. Sample $\rho^{i,j,k}$ and compute $\mathsf{ot}_2^{i,j,k} \leftarrow \mathsf{OT}_2^{1,2}(\mathsf{ot}_1^{i,j,k}, (\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}); \rho^{i,j,k})$ for all $i \in [m], j \in [\ell]$ and $k \in [\lambda]$.
3. Compute $\mathsf{ot}_2 \leftarrow \mathsf{OT}_2^{\lambda,m}(\mathsf{ot}_1, \{s_j\}_{j\in[m]})$.
4. Send ot_2 and $\{\mathsf{ot}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to R.

Receiver:

1. Compute $\mathsf{com}_2^i \leftarrow \mathsf{ExtCom}_2(\mathsf{com}_1^i)$ for all $i \in [m]$.
2. Send $(\{\mathsf{com}_2^i\}_{i\in[m]}, f)$ to S.

Round 3.

Sender:

1. Sample $\mathsf{k}_i \leftarrow \{0,1\}^\lambda$ for all $i \in [m]$.
2. Compute $(y_1, \ldots, y_m) \leftarrow \Phi_1(y)$. We assume that $y_i \in \{0,1\}^\ell$ and we denote by $y_{i,j}$ the j-th bit of y_i for all $j \in [\ell]$.
3. Compute $\mathsf{com}_3^i := \mathsf{ExtCom}_3(1^\lambda, y_i, \mathsf{k}_i, \{\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}, \rho^{i,j,k}\}_{j\in[\ell],k\in[\lambda]}; s_i)$ for all $i \in [m]$.
4. Compute $\tilde{\Phi}_i, \{\overline{\mathsf{lab}}_0^{i,j}, \overline{\mathsf{lab}}_1^{i,j}\}_{i\in[m],j\in[\ell]}, \{\mathsf{lab}_0^{i,j,k}, \mathsf{lab}_1^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]} \leftarrow \mathsf{Garble}(1^\lambda, \Phi_{2,i}; r_i)$ for all $i \in [m]$ where $r_i := \mathsf{PRF}(\mathsf{k}_i, f)$. Here $\{\overline{\mathsf{lab}}_0^{i,j}, \overline{\mathsf{lab}}_1^{i,j}\}_{i\in[m],j\in[\ell]}$ are the input labels for the sender and $\{\mathsf{lab}_0^{i,j,k}, \mathsf{lab}_1^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ are the input labels for the receiver.
5. Sample $\mathsf{k}_0^{i,j,k}, \mathsf{k}_1^{i,j,k} \leftarrow \{0,1\}^*$ and generate $\mathsf{ct}_b^{i,j,k} := \mathsf{Enc}(\mathsf{sk}_b^{i,j,k}, \mathsf{lab}_b^{i,j,k}; \mathsf{PRF}(\mathsf{k}_b^{i,j,k}, f))$ for all $b \in \{0,1\}, i \in [m], j \in [\ell]$ and $k \in [\lambda]$.
6. Send $\{\mathsf{com}_3^i, \tilde{\Phi}_i\}_{i\in[m]}$ and $\{\mathsf{ct}_0^{i,j,k}, \mathsf{ct}_1^{i,j,k}, \overline{\mathsf{lab}}_{y_{i,j}}^{i,j}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to R.

Receiver:

1. Compute $\mathsf{ot}_3^{i,j,k} \leftarrow \mathsf{OT}_3^{1,2}(\mathsf{ot}_2^{i,j,k}; r^{i,j,k})$ for all $i \in [m], j \in [\ell]$ and $k \in [\lambda]$.
2. Compute $\mathsf{ot}_3 \leftarrow \mathsf{OT}_3^{\lambda,m}(\mathsf{ot}_2; r)$.
3. Send ot_3 and $\{\mathsf{ot}_3^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to S.

Figure 7.2: The output computation of the 2PC protocol Π^{2PC}.

Output Computation.

Sender:

1. Verify that all the OTs are accepting, and reject if this is not the case.

Receiver:

1. Compute $\{s_i\}_{i\in K} := \mathsf{OT}_{\mathsf{out}}^{\lambda,m}(\mathsf{ot}_2; r)$
2. Compute $\mathsf{sk}_{x_{i,j,k}}^{i,j,k} := \mathsf{OT}_{\mathsf{out}}^{1,2}(\mathsf{ot}_2^{i,j,k}; r^{i,j,k})$ and $\mathsf{lab}_{x_{i,j,k}}^{i,j,k} := \mathsf{Dec}(\mathsf{sk}_{x_{i,j,k}}^{i,j,k}, \mathsf{ct}_{x_{i,j,k}}^{i,j,k})$ for all $i \in [m], j \in [\ell], k \in [\lambda]$.
3. For all $i \in K$:
   (a) Compute $(y_i, \mathsf{k}_i, \{sk_0'^{i,j,k}, sk_1'^{i,j,k}\}_{j\in[\ell]})$ from $\mathsf{com}_1^i, \mathsf{com}_3^i$ and randomness s_i.
   (b) Compute $r_i := \mathsf{PRF}(\mathsf{k}_i, f)$.
   (c) Compute $\tilde{\Phi}_i, \{\widetilde{\mathsf{lab}}_0^{i,j}, \widetilde{\mathsf{lab}}_1^{i,j}\}_{j\in[\ell]}, \{\mathsf{lab}_0'^{i,j,k}, \mathsf{lab}_1'^{i,j,k}\}_{j\in[\ell]} := \mathsf{Garble}(1^\lambda, \Phi_{2,i}; r_i)$.
   (d) Check if $\tilde{\Phi}_i$ that is received in the third round is the same as the one computed above.
   (e) Check that $\overline{\mathsf{lab}}_{y_{i,j}}^{i,j} = \widetilde{\mathsf{lab}}_{y_{i,j}}^{i,j}$ for all $j \in [\ell]$.

(f) Check that $\mathsf{sk}^{i,j,k}_{x_{i,j,k}} = \mathsf{sk}'^{i,j,k}_{x_{i,j,k}}$ for all $j \in [\ell], k \in [\lambda]$

(g) Compute $\mathsf{lab}^{i,j,k}_{1-x_{i,j,k}} := \mathsf{Dec}(\mathsf{sk}'^{i,j,k}_{1-x_{i,j,k}}, \mathsf{ct}^{i,j,k}_{1-x_{i,j,k}})$ for all $j \in [\ell], k \in [\lambda]$.

(h) Check that $\mathsf{lab}^{i,j,k}_{b} = \mathsf{lab}'^{i,j,k}_{b}$ for all $j \in [\ell], k \in [\lambda]$ and $b \in \{0, 1\}$.

(i) Check that $\mathsf{ot}^{i,j,k}_2 = \mathsf{OT}^{1,2}_2(\mathsf{ot}^{i,j,k}_1, (\mathsf{sk}^{i,j,k}_0, \mathsf{sk}^{i,j,k}_1); \rho^{i,j,k})$ for all $j \in [\ell], k \in [\lambda]$

4. If any of the above checks fails, output $\perp$.
5. For all $i \in [m]$, compute $z_i := \mathsf{Eval}(\tilde{\Phi}_i, \{\overline{\mathsf{lab}}^{i,j}_{y_{i,j}}\}_{j\in[\ell]}, \{\mathsf{lab}^{i,j,k}_{x_{i,j,k}}\}_{j\in[\ell],k\in[\lambda]})$.
6. Compute the output by running out_Φ on $\{z_i\}_{i\in[m]}$, the input x and the random tape used for generating $(x_1, \dots, x_m)$.

Theorem 7.1. *Let X_S be a high min-entropy random variable defined by a probability distribution $\mathcal{X}_S$, the protocol Π^{2PC} (Figure 7.1) is 1-rewinding secure list-simulatable against sometimes aborting adversaries with delayed function selection for NC_1 circuits for the ideal functionality $\mathcal{F}^{\mathcal{X}_S}$ (according to Definition 3.3), that makes block box use of $\mathsf{OT}^{1,2}, \mathsf{OT}^{k,\lambda}, \mathsf{ExtCom}, \mathsf{PRF}, \mathsf{SE}$, and GC.*

In the next two sections, we prove the security against malicious senders (Section 7.1) and security against (sometimes aborting) malicious receivers (Section 7.2).

7.1 Simulation based security against adversarial senders

We provide the formal description of our simulator Sim_S in Figure 7.3. We start the proof assuming that the adversarial sender provides an accepting third round with non-negligible probability, and then we elaborate on the case where the adversary aborts with overwhelming probability.

Hybrid H_0: This corresponds to the output of the real experiment, where the adversarial sender can send two-second rounds, and expect two to receive two valid third rounds (recall that we want to prove that our protocol is 1-rewind secure).

Hybrid H_1: In this hybrid, for each $i \in [m]$ we extract the value $\{y_i, \mathsf{k}_i, \{\mathsf{sk}'^{i,j,k}_0, \mathsf{sk}'^{i,j,k}_1, \rho^{i,j,k}\}_{j\in[\ell],k\in[\ell]}$ using extractable commitment scheme, and, as explained in the simulation (Figure 7.3), construct the set C and abort when $|C| > \lambda$. In Lemma 7.2 we prove that H_0 and H_1 are computationally indistinguishable and that H_1 runs in expected polynomial time. We refer to Figure 7.4 for the formal description of H_1.

Hybrid H_2: this hybrid is exactly like H_1 with the difference that the values $\{x_i\}_{i\in K}$ and the output of the receiver is computed running the simulator for Φ. More formally, if all the checks succeed, then H_2 does the following:

– Instruct Sim_Φ to adaptively corrupt the set of servers indexed by K and obtain $\{x_i\}_{i\in K}$. Perform all the checks that an honest receiver does and if any of these fails then return abort to the ideal functionality.

– For each $i \in C \cup K$, compute $z_i := \mathsf{Eval}(\tilde{\Phi}_i, \{\overline{\mathsf{lab}}^{i,j}_{y_{i,j}}\}_{j\in[\ell]}, \{\mathsf{lab}^{i,j,k}_{x_{i,j,k}}\}_{j\in[\ell],k\in[\lambda]})$.

– Send $\{z_i\}_{i\in C\cup K}$ as the second round message from the corrupted servers to Sim_Φ. If Sim_Φ instructs the honest client to abort, then the hybrid aborts.

The indistinguishability between the two hybrids comes immediately from the same arguments used in [IKSS21, Claim C.11].

Hybrid H_3: This hybrid behaves exactly like the previous one, with the difference that receiver messages for all the instances of $\mathsf{OT}^{1,2}$ are now computed using random inputs. The indistinguishability between the two hybrids comes from the B^{ot}-rewind security of $\mathsf{OT}^{1,2}$. In Lemma 7.5 we prove that H_2 and H_3 are computationally indistinguishable.

Hybrid H_4: In this hybrid, we use the simulator Sim_Φ and the values $\{x_i\}_{i\in C\cup K}$. Note that H_4 is identical to the output of the simulated experiment. In Lemma 7.6 we prove that H_3 and H_4 are statistically indistinguishable.

Figure 7.3: Simulator Sim_S against malicious senders.

Simulator Sim against S^*

Step 1 Perform the extraction procedure from the extractable commitments as described in Figure 7.4, and during the rewinds act as follows.

1. Use random inputs to compute all the messages of $\mathsf{OT}^{1,2}$.
2. Compute the messages of $\mathsf{OT}^{\lambda,m}$ as the honest receiver does using a random input $K \subset [m]$ with $|K| = \lambda$.

Step 2.

1. When the extractor returns (with over extraction) $\{y_i, \mathsf{k}_i, \{\mathsf{sk}_0'^{i,j,k}, \mathsf{sk}_1'^{i,j,k}, \rho^{i,j,k}\}_{i\in[m]}\}_{j\in[\ell],k\in[\ell]}$ run Sim_Φ on input $y_1, \ldots, y_m$ and function f. When Sim_Φ queries the ideal functionality on x_S record this query, and for each $i \in [m]$ do the following.
   (a) For each $j \in [\ell], k \in [m], b \in \{0,1\}$ compute $\mathsf{lab}_b^{i,j,k} := \mathsf{Dec}(\mathsf{sk}_b^{i,j,k}, \mathsf{ct}_b^{i,j,k})$
   (b) Compute $r_i := \mathsf{PRF}(\mathsf{k}_i, f)$.
   (c) Compute $\tilde{\Phi}_i, \{\widetilde{\mathsf{lab}}_0^{i,j}, \widetilde{\mathsf{lab}}_1^{i,j}\}_{j\in[\ell]}, \{\mathsf{lab}_0'^{i,j,k}, \mathsf{lab}_1'^{i,j,k}\}_{j\in[\ell]} := \mathsf{Garble}(1^\lambda, \Phi_{2,i}; r_i)$.
   (d) Check if $\tilde{\Phi}_i$ that is received in the third round is the same as the one computed above.
   (e) Check that $\overline{\mathsf{lab}}_{y_{i,j}}^{i,j} = \widetilde{\mathsf{lab}}_{y_{i,j}}^{i,j}$ for all $j \in [\ell]$.
   (f) Check that $\mathsf{sk}_b^{i,j,k} = \mathsf{sk}_b'^{i,j,k}$ and $\mathsf{lab}_b^{i,j,k} = \mathsf{lab}_b'^{i,j,k}$ for all $j \in [\ell], k \in [\lambda], b \in \{0,1\}$
   (g) Check that $\mathsf{ot}_2^{i,j,k} = \mathsf{OT}_2^{1,2}(\mathsf{ot}_1^{i,j,k}, (\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}); \rho^{i,j,k})$ for all $j \in [\ell], k \in [\lambda]$
   (h) If any of the checks fail, then add i to a set C (which is initially empty).
2. If $|C| > \lambda$ then send an abort to the ideal functionality.
3. If $|C| \leq \lambda$ then continues the execution and instruct Sim_Φ to adaptively corrupt the servers indexed by the set C and obtain $\{x_i\}_{i\in C}$.
4. Instruct Sim_Φ to adaptively corrupt the set of servers indexed by K and obtain $\{x_i\}_{i\in K}$. Perform all the checks that an honest receiver does and if any of these fails then return abort to the ideal functionality.
5. For each $i \in C \cup K$, compute $z_i := \mathsf{Eval}(\tilde{\Phi}_i, \{\overline{\mathsf{lab}}_{y_{i,j}}^{i,j}\}_{j\in[\ell]}, \{\mathsf{lab}_{x_{i,j,k}}^{i,j,k}\}_{j\in[\ell],k\in[\lambda]})$.
6. Send $\{z_i\}_{i\in C\cup K}$ as the second round message from the corrupted servers to Sim_Φ. If Sim_Φ instructs the honest client to abort, then the simulator sends abort to the ideal functionality. Otherwise, it asks the ideal functionality to deliver outputs to the honest receiver.

Figure 7.4: H_1

$\mathsf{M}^{\mathsf{H}_1}$

Round 1. Compute the first round of Π^{2PC} as the honest receiver, and send it to the adversary. Upon receiving the message $(\{\mathsf{com}_1^i\}_{i\in[m]})$ from $S^\star$ forward it to the left interface.

Round 2. Upon receiving $(\{\mathsf{com}_2^i\}_{i\in[m]}, f)$ on the left interface forward it to $S^\star$.

Round 3. Compute the third round as the honest receiver would do. Upon receiving the third round from $S^\star$ do the following.

- If the third round received from $S^\star$ is accepting (i.e., the honest receiver would not abort) then forward $\{\mathsf{com}_3^i\}_{i\in[m]}$ to the left interface.
- Else, send an abort message on the left interface.

H_1

1. Act as the honest receiver of ExtCom would, agains the sender $\mathsf{M}^{\mathsf{H}_1}$. If $\mathsf{M}^{\mathsf{H}_1}$ send a valid third round then continue as follows, abort otherwise and return what $\mathsf{M}^{\mathsf{H}_1}$ returns.
2. Rewind $\mathsf{M}^{\mathsf{H}_1}$, until three accepting transcripts for ExtCom have been collected such that 1) the transcripts all share the same first round and 2) each transcript is computed with respect to a different second round.

3. Run the 3-extractor of ExtCom on input these three transcripts, and when the extractor returns $\tau, \{y_i, \mathsf{k}_i, \{\mathsf{sk}_0'^{i,j,k}, \mathsf{sk}_1'^{i,j,k}, \rho^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\ell]}$ do the following.
   (a) For each $j \in [\ell], k \in [m], b \in \{0,1\}$ compute $\mathsf{lab}_b^{i,j,k} := \mathsf{Dec}(\mathsf{sk}_b^{i,j,k}, \mathsf{ct}_b^{i,j,k})$
   (b) Compute $r_i := \mathsf{PRF}(\mathsf{k}_i, f)$.
   (c) Compute $\tilde{\Phi}_i, \{\widetilde{\mathsf{lab}}_0^{i,j}, \widetilde{\mathsf{lab}}_1^{i,j}\}_{j\in[\ell]}, \{\mathsf{lab}_0'^{i,j,k}, \mathsf{lab}_1'^{i,j,k}\}_{j\in[\ell]} := \mathsf{Garble}(1^\lambda, \Phi_{2,i}; r_i)$.
   (d) Check if $\tilde{\Phi}_i$ that is received in the third round of the thread that contains τ.
   (e) Check that $\overline{\mathsf{lab}}_{y_{i,j}}^{i,j} = \widetilde{\mathsf{lab}}_{y_{i,j}}^{i,j}$ for all $j \in [\ell]$.
   (f) Check that $\mathsf{sk}_b^{i,j,k} = \mathsf{sk}_b'^{i,j,k}$ and $\mathsf{lab}_b^{i,j,k} = \mathsf{lab}_b'^{i,j,k}$ for all $j \in [\ell], k \in [\lambda], b \in \{0,1\}$
   (g) Check that $\mathsf{ot}_2^{i,j,k} = \mathsf{OT}_2^{1,2}(\mathsf{ot}_1^{i,j,k}, (\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}); \rho^{i,j,k})$ for all $j \in [\ell], k \in [\lambda]$
   (h) If any of the checks fail, then add i to a set C (which is initially empty).
4. If $|C| > \lambda$ then abort.
5. If $|C| \leq \lambda$ the continue as follows.
6. For all $i \in [m]$, compute $z_i := \mathsf{Eval}(\tilde{\Phi}_i, \{\overline{\mathsf{lab}}_{y_{i,j}}^{i,j}\}_{j\in[\ell]}, \{\mathsf{lab}_{x_{i,j,k}}^{i,j,k}\}_{j\in[\ell],k\in[\lambda]})$.
7. Compute the output by running out_Φ on $\{z_i\}_{i\in[m]}$, the input x and the random tape used for generating $(x_1, \dots, x_m)$.

Lemma 7.2. *Assuming that* $\mathsf{OT}^{\lambda,m}$ *is* B^{ot}*-rewind secure, private against corrupted sender k-out-of-n oblivious transfer protocol, then* H_0 *and* H_1 *are computationally indistinguishable.*

Proof. Let $\{y_\alpha, \mathsf{k}_\alpha, \{\mathsf{sk}_0'^{\alpha,j,k}, \mathsf{sk}_1'^{\alpha,j,k}, \rho^{\alpha,j,k}\}_{\alpha\in[m],j\in[\ell],k\in[\ell]}$ be the value extracted from the α-th extractable commitment in H_1. For each $j \in [\ell], k \in [m], b \in \{0,1\}$ let $\mathsf{lab}_b^{\alpha,j,k}$ be the value obtained from $\mathsf{Dec}(\mathsf{sk}_b^{\alpha,j,k}, \mathsf{ct}_b^{\alpha,j,k})$. Let $r_\alpha := \mathsf{PRF}(\mathsf{k}_\alpha, f)$ and $\tilde{\Phi}_{2,\alpha}, \{\widetilde{\mathsf{lab}}_0^{\alpha,j}, \widetilde{\mathsf{lab}}_1^{\alpha,j}\}_{j\in[\ell]}, \{\mathsf{lab}_0'^{\alpha,j,k}, \mathsf{lab}_1'^{\alpha,j,k}\}_{j\in[\ell]} := \mathsf{Garble}(1^\lambda, \Phi_{2,\alpha}; r_\alpha))$.

We denote with Ext_α the event where the following occurs:

1. $\tilde{\Phi}_{2,\alpha}$ is received in the third round of the thread that contains τ.
2. $\overline{\mathsf{lab}}_{y_{\alpha,j}}^{\alpha,j} = \widetilde{\mathsf{lab}}_{y_{\alpha,j}}^{\alpha,j}$ for all $j \in [\ell]$.
3. $\mathsf{sk}_b^{\alpha,j,k} = \mathsf{sk}_b'^{\alpha,j,k}$ for all $j \in [\ell], k \in [\lambda], b \in \{0,1\}$
4. $\mathsf{lab}_b^{\alpha,j,k} = \mathsf{lab}_b'^{\alpha,j,k}$ for all $j \in [\ell], k \in [\lambda]$ and $b \in \{0,1\}$.
5. $\mathsf{ot}_2^{\alpha,j,k} = \mathsf{OT}_2^{1,2}(\mathsf{ot}_1^{\alpha,j,k}, (\mathsf{sk}_0^{\alpha,j,k}, \mathsf{sk}_1^{\alpha,j,k}); \rho^{\alpha,j,k})$ for all $j \in [\ell], k \in [\lambda]$

We prove the following intermediate claim.

Claim 7.3 *Let ε be the aborting probability of the adversary, then* $\forall \alpha \in [K], \Pr[\,\mathsf{Ext}_\alpha | \alpha \in K\,] = 1 - \text{negl}(\lambda)$*, where K denotes the input used to generate the messages of* $\mathsf{OT}^{\lambda,m}$ *in* H_1*.*

Proof. The proof follows from the observation that during the rewinds H_1 keeps fixed the input K used $\mathsf{OT}^{\lambda,m}$. Hence, in any triple of accepting transcripts, the extractable commitments indexed by α ($\forall \alpha \in K$), are honestly generated (this can be checked upon receiving the output of $\mathsf{OT}^{\lambda,m}$). Hence, we can invoke the 3-extractability property and be sure that we have extracted corresponds to the correct message.

□

Claim 7.4 $\forall \alpha \in [m], |\Pr[\,\mathsf{Ext}_\alpha | \alpha \in K\,] - \Pr[\,\mathsf{Ext}_\alpha | \alpha \notin K\,]| \leq \text{negl}(\lambda)$*, where K denotes the input used to generate the messages of* $\mathsf{OT}^{\lambda,m}$ *in* H_1*.*

Proof. In Claim 7.3 we have argued that the extraction from the extractable commitments indexed by α is successful for every $\alpha \in K$. We now argue that we can extract successfully from the α-th commitment even when $\alpha \notin K$. Suppose by contradiction that the claim does not hold. This means that the probability of success of the extractor on the α-th commitment depends on the input K used by the receiver to compute the messages of $\mathsf{OT}^{\lambda,m}$. This intuitively means that we can construct a reduction to the B^{ot}-rewind receiver security of $\mathsf{OT}^{\lambda,m}$. The adversary $\mathcal{A}^{\mathsf{OT}^{\lambda,m}}$, on input $K_0, K_1 \subset [m]$, which denote the challenge messages, such that $\alpha \in K_0$ and $\alpha \notin K_1$, interacts with a challenger. The challenger that acts as the honest receiver for $\mathsf{OT}^{\lambda,m}$ using as input K_b, with $b \leftarrow \{0,1\}$. $\mathcal{A}^{\mathsf{OT}^{\lambda,m}}$ works as follows.

1. Upon receiving ot_1 from the external challenger, compute the first round as the honest receiver would in the real-world experiment, but use ot_1, in the place of the message that would have been generated by running $\mathsf{OT}^{k,\lambda}$. Send the first round of Π^{2PC} to $S^\star$.
2. Upon receiving the first round from $S^\star$, reply with a second round computed as the honest receiver would do.
3. Upon receiving two second rounds from $S^\star$, forward $(\mathsf{ot}_2^1, \mathsf{ot}_2^2)$ to the external challenger (where ot_2^d represents the sender message for $\mathsf{OT}^{\lambda,m}$ send in the d-th second round message received from $S^\star$).
4. Upon receiving $(\mathsf{ot}_3^1, \mathsf{ot}_3^2)$ from the challenger use these to compute two accepting third rounds for Π^{2PC}.
5. Upon receiving the third round of Π^{2PC} from $S^\star$, rewind $S^\star$ collect the transcript for the α-th extractable commitment and repeat the steps $2, 3$ and 4 twice.
6. If three transcripts for the α extractable commitments have not been obtained, then return a random guess, else rely on the 3-extractability property of ExtCom to obtain $\{y_\alpha, \mathsf{k}_\alpha, \{\mathsf{sk}_0'^{\alpha,j,k}, \mathsf{sk}_1'^{\alpha,j,k}, \rho^{\alpha,j,k}\}_{j\in[\ell],k\in[\lambda]}$.
7. For each $j \in [\ell], k \in [m], b' \in \{0,1\}$ let $\mathsf{lab}_{b'}^{i,j,k}$ be the value obtained from $\mathsf{Dec}(\mathsf{sk}_{b'}^{\alpha,j,k}, \mathsf{ct}_{b'}^{\alpha,j,k})$. Let $r_\alpha := \mathsf{PRF}(\mathsf{k}_\alpha, f)$ and $\tilde{\Phi}_{2,\alpha}, \{\widetilde{\mathsf{lab}}_0^{\alpha,j}, \widetilde{\mathsf{lab}}_1^{\alpha,j}\}_{j\in[\ell]}, \{\mathsf{lab}_0'^{\alpha,j,k}, \mathsf{lab}_1'^{\alpha,j,k}\}_{j\in[\ell]} := \mathsf{Garble}(1^\lambda, \Phi_{2,\alpha}; r_\alpha))$. Return 1 if all the following conditions are satisfied with respect to the transcript generated in the main thread, return 0 otherwise
   (a) $\tilde{\Phi}_{2,\alpha}$ is received in the third round of the main thread.
   (b) $\overline{\mathsf{lab}}_{y_{\alpha,j}}^{\alpha,j} = \widetilde{\mathsf{lab}}_{y_{\alpha,j}}^{\alpha,j}$ for all $j \in [\ell]$.
   (c) $\mathsf{sk}_{b'}^{\alpha,j,k} = \mathsf{sk}_{b'}'^{\alpha,j,k}$ for all $j \in [\ell], k \in [\lambda], b' \in \{0,1\}$
   (d) $\mathsf{lab}_{b'}^{\alpha,j,k} = \mathsf{lab}_{b'}'^{\alpha,j,k}$ for all $j \in [\ell], k \in [\lambda]$ and $b' \in \{0,1\}$.
   (e) $\mathsf{ot}_2^{\alpha,j,k} = \mathsf{OT}_2^{1,2}(\mathsf{ot}_1^{\alpha,j,k}, (\mathsf{sk}_0^{\alpha,j,k}, \mathsf{sk}_1^{\alpha,j,k}); \rho^{\alpha,j,k})$ for all $j \in [\ell], k \in [\lambda]$

We note that the rewinds performed during the extraction do not perturb the reduction due to the B^{ot} rewind security of $\mathsf{OT}^{\lambda,m}$ (we recall that $B^{\mathsf{ot}} = 6$).

Let ε be the probability that $S^\star$ provides an accepting transcript in H_1. Then we have that $\Pr[\mathsf{Ext}_\alpha | \alpha \in K] = \Pr\left[\mathcal{A}^{\mathsf{OT}^{\lambda,m}} = 1 | b = 0\right] \cdot \varepsilon^3$ and $\Pr[\mathsf{Ext}_\alpha | \alpha \notin K] = \Pr\left[\mathcal{A}^{\mathsf{OT}^{\lambda,m}} = 1 | b = 1\right] \cdot \varepsilon^3$. By contradiction, it must be that $|\Pr\left[\mathcal{A}^{\mathsf{OT}^{\lambda,m}} = 1 | b = 0\right] - \Pr\left[\mathcal{A}^{\mathsf{OT}^{\lambda,m}} = 1 | b = 1\right]|$ is non-negligible, but this contradicts the receiver privacy of $\mathsf{OT}^{\lambda,m}$. □

We are now ready to prove that H_0 and H_1 are computationally indistinguishable. We first argue that for each $i \in C \cap K$ one of the checks done by the honest receiver in H_1 independently fails with probability at least 1/2. To see why this is the case, consider some $i \in K \cap C$. If i was added to C as a result of the check in Step 2.(c) failing, then the honest receiver also catches this with probability 1. If i was added to C as a result of a check in Step 2.(e) fails, then it means that there exists j, k and $b \in \{0,1\}$ such that $\mathsf{sk}_b'^{i,j,k} \neq \mathsf{sk}_b^{i,j,k}$. In this case, the honest receiver catches this with probability at least 1/2 since $x_{i,j,k} = b$ with probability 1/2. If i was added to C as a result of the check in Step 2.(d) failing, then honest receiver in H_1 catches this with probability 1. This shows that for each $i \in C \cap K$, one of the checks done by the honest receiver in H_1 independently fails with probability at least 1/2.

We have proven in Theorem 7.4 that the probability of extracting successfully from the α-th commitment is independent of whether α belongs to K or not. Hence, we can now prove that if K is chosen uniformly at random, then the probability that $|K \cap C| \leq \lambda/100$ is $2^{-O(\lambda)}$. This last part of the proof follows from exactly the same arguments of [IKSS21, Claim C.10.]

Finally, we need to prove that H_1 terminates in expected polynomial time. We note that the rewinding threads are identically distributed to the main thread in H_1. Let ϵ denote the probability that $\mathcal{A}$ does not abort. Then, we have that the expected running time of H_1 is $1 - \epsilon + \epsilon\frac{1}{\epsilon}$ times a fixed polynomial required to compute the commitments and the OT messages. Thus, the overall expected cost is $\text{poly}(\lambda)(2 - \epsilon)$.

□

Lemma 7.5. *Assuming that* $\mathsf{OT}^{1,2}$ *is* B^{ot}*-rewind secure, private against corrupted sender* 1*-out-of-*2 *oblivious transfer protocol, then* H_2 *and* H_3 *are computationally indistinguishable.*

Proof. Assume by contradiction that there exists a distinguisher that can distinguish between the two hybrids with non-negligible advantage. We show how to construct an adversary $\mathcal{A}^{\mathsf{OT}^{1,2}}$ that breaks the receiver privacy of $\mathsf{OT}^{1,2}$. The OT challenger receives challenge messages $\{x_{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ and 0. That is, the challenger, in the (i,j,k)-th OT execution will either use the input $x_{i,j,k}$ or 0. $\mathcal{A}^{\mathsf{OT}^{1,2}}$ works as follows.

– Upon receiving the OT messages from the challenger, use them to compute a complete interaction with $S^\star$.
– Upon receiving the third round, rewind $S^\star$ two times and use the 3-extractability property of ExtCom thus obtaining $\{y_i, \mathsf{k}_i, \{\mathsf{sk}_0'^{i,j,k}, \mathsf{sk}_1'^{i,j,k}, \rho^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\ell]}$, and construct the set C as described in H_2 (and H_3). If $C > \lambda$ then return a random guess, return whether $S^\star$ returns.

Note that when the OT messages are computed using the values $\{x_{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$, then the view of $S^\star$ corresponds to H_2, else it corresponds to H_3. Moreover, the extraction of $\{y_i, \mathsf{k}_i, \{\mathsf{sk}_0'^{i,j,k}, \mathsf{sk}_1'^{i,j,k}, \rho^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\ell]}$ (hence the reduction) is successful with probability ε^3, where ε is the non-negligible probability that $S^\star$ provides an accepting transcript. The proof ends with the observation that the rewinds performed to extract from ExtCom do not perturb the reduction due to the B^{ot} rewind security (we recall that $B^{\mathsf{ot}} = 6$, and that in each thread $S^\star$ can ask to different second rounds).

□

Lemma 7.6. *Assuming the security of the outer MPC protocol, we have* H_3 *is statistically indistinguishable from* H_4.

Proof. Assume by contradiction that there exists a distinguisher that can distinguish between the two hybrids with non-negligible advantage. We show how to construct an adversary $\mathcal{A}^\Phi$ that breaks the security of the protocol Φ. $\mathcal{A}^\Phi$ works as follows.

– Rewind the adversarial sender two times and use the 3-extractability property of ExtCom thus obtaining $\{y_i, \mathsf{k}_i, \{\mathsf{sk}_0'^{i,j,k}, \mathsf{sk}_1'^{i,j,k}, \rho^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\ell]}$, and construct the set C as described in H_3 (and H_4). If $C > \lambda$ then return a random guess, otherwise continue as follows.
– Send $\{y_i\}_{i\in[m]}$ to the external challenger.
– Corrupt the sender client and sends x_R as the input of the receiver client to the external challenger.
– Before receiving the second round message from the challenger, ask the challenger to adaptively corrupt the servers indexed by the set C and the set K.
– Upon receiving $\{x_i\}_{i\in C\cup K}$, use it to to perform all the checks as in H_3 (and H_4) and evaluates the garbled circuits $\tilde{\Phi}_i$ for each $i \in C \cup K$ thus obtaining $\{z_i\}_{i\in C\cup K}$.
– Send $\{z_i\}_{i\in C\cup K}$ to the challenger and obtain the output of the honest receiver.
– Return whatever the adversarial sender returns.

Note that if the outer protocol messages given by the challenger are generated using the real algorithms then the view of $S^\star$ is identical to the view it has in H_3. Otherwise, it is identical to the view $S^\star$ has in H_4. The proof ends with the observation that the extraction of $\{y_i, \mathsf{k}_i, \{\mathsf{sk}_0'^{i,j,k}, \mathsf{sk}_1'^{i,j,k}, \rho^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\ell]}$ (hence the reduction) is successful with probability ε^3, if ε is the non-negligible probability that $S^\star$ provides an accepting transcript.

□

7.2 Proof against corrupted receivers

We recall that we prove the security of the protocol only against non-aborting receivers. We denote the simulator of $\mathsf{OT}^{\lambda,m}$ with $(\mathsf{Sim}_1^{\lambda,m}, \mathsf{Sim}_2^{\lambda,m})$ and the simulator of $\mathsf{OT}^{\lambda,m}$ with $(\mathsf{Sim}_1^{1,2}, \mathsf{Sim}_2^{1,2})$.

We consider the following sequence of hybrid experiments.

Hybrid H_0: this is identical to the real game experiment.

Hybrid H_1: The difference between this and the previous hybrid experiment is that $\mathsf{Sim}_1^{1,2}$ and $\mathsf{Sim}_1^{\lambda,m}$ are run. Upon receiving the output from these simulators, the experiment rewinds up to the second round, and completes an execution against the corrupted receiver as the honest sender would do.
H_0 and H_1 are statistical indistinguishable due to the security of $\mathsf{Sim}_1^{1,2}$ and $\mathsf{Sim}_1^{\lambda,m}$. We recall that this holds due to the fact that these simulators simply perform a sequence of rewinds, where each rewinding thread is identical to the main thread in the real-world experiment.

Hybrid H_2: The difference with the previous hybrid is that upon receiving the outputs from $\mathsf{Sim}_1^{1,2}$ and $\mathsf{Sim}_1^{\lambda,m}$, these are used to set the inputs (and run) $\mathsf{Sim}_2^{1,2}$ and $\mathsf{Sim}_2^{\lambda,m}$ respectively. We refer to Figure 7.5 for the formal description of H_2. The indistinguishability between the two hybrids comes from the security offered by $\mathsf{Sim}_2^{1,2}$ and $\mathsf{Sim}_2^{\lambda,m}$.

Hybrid H_3^{γ}, with $\gamma \in \{1, \dots, \lambda\}$: Let K be the valued returned by $\mathsf{Sim}_1^{\lambda,m}$. This hybrid works exactly like the previous one, with the difference that, when the extraction performed by $\mathsf{Sim}_1^{1,2}$ and $\mathsf{Sim}_1^{\lambda,m}$ is successful, then the j-th execution of ExtCom_1 commits to 0^λ if $i \in I$ where I is a (uniform-)random subset of $[m] \setminus K$ with $|I| = \gamma$. We refer to Figure 7.6 for the formal description of the hybrid. We note that H_3^0 and H_2 are identical. In Lemma 7.7 we prove that $\mathsf{H}_3^{\gamma-1} \approx \mathsf{H}_3^{\gamma}$ for each $\gamma \in \{2, \dots, \lambda\}$.

Hybrid H_4: The difference between this and the hybrid H_3^{λ} is that any PRF evaluation is replaced by the evaluation of a random function (we refer to Figure 7.7 for the formal description of the hybrid). The indistinguishability between this hybrid and H_3^{λ} comes from the security of the PRF.

Hybrid H_5: The difference between this and the hybrid H_4^{λ} is that during the phase denoted as *forcing the output*, all the ciphertexts that are not opened to the adversarial receiver (this is non-ambiguously defined by the values extracted using the OT simulators) are set to be encryption of a constant value (0^λ in our case). We refer to Figure 7.8 for the formal description of the hybrid. The indistinguishability between this hybrid and H_4 comes from the semantic security of the encryption scheme.

Hybrid H_6: The difference between this and the hybrid H_5 is that during the phase denoted as *forcing the output*, all the garbled circuits not open to the adversarial receiver (this is non-ambiguously defined by the values extracted using the OT simulators of $\mathsf{OT}^{\lambda,m}$) are simulated. We refer to Figure 7.9 for the formal description of the hybrid. The indistinguishability between this hybrid and H_5 comes from the security of the garbled circuit.

Hybrid H_7: The difference between this and the hybrid H_6^{λ} is that during the phase denoted as *forcing the output*, we use the simulator Sim_Φ generates the first round messages $\{y_i\}_{i \in K}$ from honest sender as well as the second round messages $\{z_i\}_{i \notin K}$ from the honest servers. We refer to Figure 7.10 for the formal description of the hybrid. The indistinguishability between this hybrid and H_6 comes from the security of the outer protocol Φ.
This part of the proof ends with the observation that the H_7 corresponds to the simulated experiment (see Figure 7.11). In the case where the sender aborts with overwhelming probability, the proof follows exactly the same steps, with the exception that there is no need to extract and to argue that the extraction is correct. Indeed in this case the honest receiver would simply abort.
We finally argue that the running time of the simulator is expected polynomial time. This holds since the work performed in each rewinding thread is polynomial, and moreover, the rewinding threads are identically distributed to the simulated thread. Given that the simulator just needs to collect three accepting transcripts, we have that in expectation the simulator will run in polynomial time.

Figure 7.5: H_2

Input extraction. Run $\mathsf{Sim}_1^{1,2}$ and $\mathsf{Sim}_1^{\lambda,m}$. For each rewind performed by the simulators prior to the extraction do the following

1. Let $\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}$ be the input sampled by $\mathsf{Sim}_1^{1,2}$ from $D^{1,2}$, for each $i \in [m], j \in [\ell], k \in [\lambda]$, and let $s_1, \dots, s_m$ the values that $\mathsf{Sim}_1^{\lambda,m}$ samples when querying $D^{\lambda,m}$.
2. Compute $\mathsf{com}_1^i := \mathsf{ExtCom}_1(1^\lambda; s_i)$ for all $i \in [m]$ and send $(\{\mathsf{com}_1^i\}_{i \in [m]})$ to $R^\star$.
3. Upon receiving ot_1^i and $\{\mathsf{ot}_1^{i,j,k}\}_{i \in [m], j \in [\ell], k \in [\lambda]}$ from $R^\star$ forward these to $\mathsf{Sim}_1^{\lambda,m}$ and $\mathsf{Sim}_1^{1,2}$ respectively.

4. When $\mathsf{Sim}_1^{1,2}$ or $\mathsf{Sim}_1^{\lambda,m}$ send $\{\mathsf{ot}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ and ot_2 respectively, forward these messages to $R^\star$.
5. Upon receiving two second-round $\{\mathsf{c},\mathsf{c}'\}$ from $R^\star$, sample $x_S \leftarrow D_S$ and for each $(\{\mathsf{com}_2^i\}_{i\in[m]}, f) \in \{\mathsf{c},\mathsf{c}'\}$ do the following
   - For each $i \in [m]$
     i. Sample $\mathsf{k}_i \leftarrow \{0,1\}^\lambda$ for all $i \in [m]$.
     ii. Compute $(y_1,\ldots,y_m) \leftarrow \Phi(x_S)$.
     iii. Compute $\mathsf{com}_3^i := \mathsf{ExtCom}_3(1^\lambda, y_i, \mathsf{k}_i, \{\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}, \rho^{i,j,k}\}_{j\in[\ell],k\in[\lambda]}; s_i)$ for all $i \in [m]$ (where $\rho^{i,j,k}$ is the randomness used by $\mathsf{Sim}_1^{1,2}$ to compute the message $\mathsf{ot}_2^{i,j,k}$).
     iv. Compute $\tilde{\Phi}_i, \{\overline{\mathsf{lab}}_0^{i,j}, \overline{\mathsf{lab}}_1^{i,j}\}_{i\in[m],j\in[\ell]}, \{\mathsf{lab}_0^{i,j,k}, \mathsf{lab}_1^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]} \leftarrow \mathsf{Garble}(1^\lambda, \Phi_{2,i}, r_i)$ for all $i \in [m]$ where $r_i := \mathsf{PRF}(\mathsf{k}_i, f)$. Here $\{\overline{\mathsf{lab}}_0^{i,j}, \overline{\mathsf{lab}}_1^{i,j}\}_{i\in[m],j\in[\ell]}$ are the input labels for the sender and $\{\mathsf{lab}_0^{i,j,k}, \mathsf{lab}_1^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ are the input labels for the receiver.
     v. Generate $\mathsf{ct}_b^{i,j,k} := \mathsf{Enc}(\mathsf{sk}_b^{i,j,k}, \mathsf{lab}_b^{i,j,k}; \mathsf{PRF}(\mathsf{k}_b^{i,j,k}, f))$ for all $b \in \{0,1\}, i \in [m], j \in [\ell]$ and $k \in [\lambda]$.
   - Send $\tilde{C}, \{\mathsf{ct}_0^i, \mathsf{ct}_1^i, \mathsf{com}_3^i, \tilde{\Phi}_i\}_{i\in[m]}$ and $\{\mathsf{ct}_0^{i,j,k}, \mathsf{ct}_1^{i,j,k}, \overline{\mathsf{lab}}_{y_{i,j}}^{i,j}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to $R^\star$.
6. Upon receiving ot_3 and $\{\mathsf{ot}_3^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ forward these to $\mathsf{Sim}_1^{\lambda,m}$ and $\mathsf{Sim}_1^{1,2}$ respectively.

Forcing the output. Upon receiving $(1^{\kappa_1}, \{x_{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}\}, z_1)$ from $\mathsf{Sim}_1^{1,2}$ and $(1^{\kappa_2}, K, z_2)$ from $\mathsf{Sim}_1^{\lambda,m}$ do the following.
1. Let $\kappa \leftarrow \min\{\kappa_1,\kappa_2\}$, for each $\iota \in [\kappa]$, $i \in [m], j \in [\ell], k \in [\lambda]$, sample $\mathsf{sk}_\iota^{i,j,k} \leftarrow \{0,1\}^\lambda$, and set $(s_\iota^1,\ldots,s_\iota^m) := (s^1,\ldots,s^m)$.
2. For each $\iota \in [\kappa]$ (i.e., for each rewind performed by the OT simulators $\mathsf{Sim}_2^{1,2}$ and $\mathsf{Sim}_2^{\lambda,m}$ on input respectively $(1^\kappa, \{\mathsf{sk}_\iota^{i,j,k}\}_{\iota\in[\kappa],i\in[m],j\in[\ell],k\in[\lambda]}, z_1)$ and $(\{s_\iota^i\}_{\iota\in\kappa,i\in K}), z_2)$, receive ot_2 and $\{\mathsf{ot}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ from $\mathsf{Sim}_2^{\lambda,m}$ and $\mathsf{Sim}_2^{1,2}$ respectively, and do the following.
   (a) For each $i \in [m]$ pick $\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k} \leftarrow \{0,1\}^\lambda$ for each $j \in [\ell]$, $k \in [\lambda]$
   (b) For each $i \in K$ compute $\mathsf{ot}_2^{i,j,k} \leftarrow \mathsf{OT}_2^{1,2}(\mathsf{ot}_1^{i,j,k}, (\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}); \rho^{i,j,k})$ for all $j \in [\ell]$ and $k \in [\lambda]$.
   (c) Send ot_2 and $\{\mathsf{ot}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to $R^\star$.
3. Upon receiving two second-round $\{\mathsf{c},\mathsf{c}'\}$ from $R^\star$, sample $x_S \leftarrow D_S$ and for each $(\{\mathsf{com}_2^i\}_{i\in[m]}, f) \in \{\mathsf{c},\mathsf{c}'\}$ do the following
   - For each $b \in \{0,1\}$, for each $i \in [m]$ do the following
     i. Compute $(y_1,\ldots,y_m) \leftarrow \Phi_1(x_S)$.
     ii. $r_i := \mathsf{PRF}(\mathsf{k}_i, f)$.
     iii. $\tilde{\Phi}_i, \{\overline{\mathsf{lab}}_0^{i,j}, \overline{\mathsf{lab}}_1^{i,j}\}_{i\in[m],j\in[\ell]}, \{\mathsf{lab}_0^{i,j,k}, \mathsf{lab}_1^{i,j,k}\}_{j\in[\ell],k\in[\lambda]} \leftarrow \mathsf{Garble}(1^\lambda, \Phi_{2,i}; r_i)$.
     iv. Compute $\mathsf{com}_3^i := \mathsf{ExtCom}_3(1^\lambda, y, \mathsf{k}_i, \{\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}, \rho^{i,j,k}\}_{j\in[\ell],k\in[\lambda]}; s_i)$,
     v. Sample $\mathsf{k}_0^{i,j,k}, \mathsf{k}_1^{i,j,k} \leftarrow \{0,1\}^*$ and generate $\mathsf{ct}_0^{i,j,k} := \mathsf{Enc}(\mathsf{sk}_0^{i,j,k}, \mathsf{lab}_0^{i,j,k}; \mathsf{PRF}(\mathsf{k}_0^{i,j,k},))$, $\mathsf{ct}_1^{i,j,k} := \mathsf{Enc}(\mathsf{sk}_1^{i,j,k}, \mathsf{lab}_1^{i,j,k}; \mathsf{PRF}(\mathsf{k}_1^{i,j,k},))$ for all $k \in [\lambda]$.
   - Send $\{\mathsf{com}_3^i, \tilde{\Phi}_i\}_{i\in[m]}$ and $\{\mathsf{ct}_0^{i,j,k}, \mathsf{ct}_1^{i,j,k}, \overline{\mathsf{lab}}_{y_{i,j}}^{i,j}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$

Figure 7.6: Hybrid H_3^γ with $\gamma \in [\lambda]$.

Input extraction. Run $\mathsf{Sim}_1^{1,2}$ and $\mathsf{Sim}_1^{\lambda,m}$. For each rewind performed by the simulators prior to the extraction do the following
1. Let $\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}$ be the input sampled by $\mathsf{Sim}_1^{1,2}$ from $D^{1,2}$, for each $i \in [m], j \in [\ell], k \in [\lambda]$, and let $s_1,\ldots,s_m$ the values that $\mathsf{Sim}_1^{\lambda,m}$ samples when querying $D^{\lambda,m}$.
2. Sample $x_S \leftarrow D_S$.

3. Compute $\mathsf{com}_1^i := \mathsf{ExtCom}_1(1^\lambda; s_i)$ for all $i \in [m]$ and send $(\{\mathsf{com}_1^i\}_{i\in[m]})$ to $R^\star$.
4. Upon receiving ot_1^i and $\{\mathsf{ot}_1^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ from $R^\star$ forward these to $\mathsf{Sim}_1^{\lambda,m}$ and $\mathsf{Sim}_1^{1,2}$ respectively.
5. When $\mathsf{Sim}_1^{1,2}$ or $\mathsf{Sim}_1^{\lambda,m}$ send $\{\mathsf{ot}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ and ot_2 respectively, forward these messages to $R^\star$.
6. Upon receiving two second-round $\{\mathsf{c}, \mathsf{c}'\}$ from $R^\star$, sample $x_S \leftarrow D_S$ and for each $(\{\mathsf{com}_2^i\}_{i\in[m]}, f) \in \{\mathsf{c}, \mathsf{c}'\}$ do the following
   - For each $i \in [m]$ do the following
     i. Sample $\mathsf{k}_i \leftarrow \{0,1\}^\lambda$ for all $i \in [m]$.
     ii. Compute $(y_1, \ldots, y_m) \leftarrow \Phi_1(x_S)$.
     iii. Compute $\mathsf{com}_3^i := \mathsf{ExtCom}_3(1^\lambda, y_i, \mathsf{k}_i, \{\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}, \rho^{i,j,k}\}_{j\in[\ell],k\in[\lambda]}; s_i)$ for all $i \in [m]$ (where $\rho^{i,j,k}$ is the randomness used by $\mathsf{Sim}_1^{1,2}$ to compute the message $\mathsf{ot}_2^{i,j,k}$).
     iv. Compute $\tilde{\Phi}_i, \{\overline{\mathsf{lab}}_0^{i,j}, \overline{\mathsf{lab}}_1^{i,j}\}_{i\in[m],j\in[\ell]}, \{\mathsf{lab}_0^{i,j,k}, \mathsf{lab}_1^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]} \leftarrow \mathsf{Garble}(1^\lambda, \Phi_{2,i}, r_i)$ for all $i \in [m]$ where $r_i := \mathsf{PRF}(\mathsf{k}_i, f)$. Here $\{\overline{\mathsf{lab}}_0^{i,j}, \overline{\mathsf{lab}}_1^{i,j}\}_{i\in[m],j\in[\ell]}$ are the input labels for the sender and $\{\mathsf{lab}_0^{i,j,k}, \mathsf{lab}_1^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ are the input labels for the receiver.
     v. Generate $\mathsf{ct}_b^{i,j,k} := \mathsf{Enc}(\mathsf{sk}_b^{i,j,k}, \mathsf{lab}_b^{i,j,k}; \mathsf{PRF}(\mathsf{k}_b^{i,j,k}, f))$ for all $b \in \{0,1\}, i \in [m], j \in [\ell]$ and $k \in [\lambda]$.
   (a) Send $\tilde{C}, \{\mathsf{ct}_0^i, \mathsf{ct}_1^i, \mathsf{com}_3^i, \tilde{\Phi}_i\}_{i\in[m]}$ and $\{\mathsf{ct}_0^{i,j,k}, \mathsf{ct}_1^{i,j,k}, \overline{\mathsf{lab}}_{y_{i,j}}^{i,j}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to $R^\star$.
7. Upon receiving ot_3 and $\{\mathsf{ot}_3^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ forward these to $\mathsf{Sim}_1^{\lambda,m}$ and $\mathsf{Sim}_1^{1,2}$ respectively.

Forcing the output. Upon receiving $(1^{\kappa_1}, \{x_{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}\}, z_1)$ from $\mathsf{Sim}_1^{1,2}$ and $(1^{\kappa_2}, K, z_2)$ from $\mathsf{Sim}_1^{\lambda,m}$ do the following.

1. Let $\kappa \leftarrow \min\{\kappa_1, \kappa_2\}$, for each $\iota \in [\kappa]$, $i \in [m], j \in [\ell], k \in [\lambda]$, sample $\mathsf{sk}_\iota^{i,j,k} \leftarrow \{0,1\}^\lambda$. Let I be a random subset of $[m] \setminus K$ with $|I| = \alpha$, for each $i \in I$, set $s_\iota^i := 0^\lambda$, else set $s_\iota^i := s^i$.
2. For each $\iota \in [\kappa]$ (i.e., for each rewind performed by the OT simulators $\mathsf{Sim}_2^{1,2}$ and $\mathsf{Sim}_2^{\lambda,m}$ on input respectively $(\{\mathsf{sk}_\iota^{i,j,k}\}_{\iota\in[\kappa],i\in[m],j\in[\ell],k\in[\lambda]}, z_1)$ and $(\{s_\iota^i\}_{\iota\in\kappa,i\in K}, z_2)$, receive ot_2 and $\{\mathsf{ot}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ from $\mathsf{Sim}_2^{\lambda,m}$ and $\mathsf{Sim}_2^{1,2}$ respectively, and do the following.
   (a) For each $i \in [m]$ pick $\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k} \leftarrow \{0,1\}^\lambda$ for each $j \in [\ell]$, $k \in [\lambda]$
   (b) For each $i \in K$ compute $\mathsf{ot}_2^{i,j,k} \leftarrow \mathsf{OT}_2^{1,2}(\mathsf{ot}_1^{i,j,k}, (\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}); \rho^{i,j,k})$ for all $j \in [\ell]$ and $k \in [\lambda]$.
   (c) Send ot_2 and $\{\mathsf{ot}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to $R^\star$.
3. Upon receiving two second-round $\{\mathsf{c}, \mathsf{c}'\}$ from $R^\star$, sample $x_S \leftarrow D_S$ and for each $(\{\mathsf{com}_2^i\}_{i\in[m]}, f) \in \{\mathsf{c}, \mathsf{c}'\}$ do the following
   - For each $i \in [m]$ do the following
     i. Pick $\mathsf{k}_i \leftarrow \{0,1\}^\lambda$.
     ii. Compute $(y_1, \ldots, y_m) \leftarrow \Phi_1(x_S)$.
     iii. $r_i := \mathsf{PRF}(\mathsf{k}_i, f)$.
     iv. $\tilde{\Phi}_i, \{\overline{\mathsf{lab}}_0^{i,j}, \overline{\mathsf{lab}}_1^{i,j}\}_{i\in[m],j\in[\ell]}, \{\mathsf{lab}_0^{i,j,k}, \mathsf{lab}_1^{i,j,k}\}_{j\in[\ell],k\in[\lambda]} \leftarrow \mathsf{Garble}(1^\lambda, \Phi_{2,i}; r_i)$.
     v. Sample $\mathsf{k}_0^{i,j,k}, \mathsf{k}_1^{i,j,k} \leftarrow \{0,1\}^*$ and generate $\mathsf{ct}_0^{i,j,k} := \mathsf{Enc}(\mathsf{sk}_0^{i,j,k}, \mathsf{lab}_0^{i,j,k}; \mathsf{PRF}(\mathsf{k}_0^{i,j,k},))$, $\mathsf{ct}_1^{i,j,k} := \mathsf{Enc}(\mathsf{sk}_1^{i,j,k}, \mathsf{lab}_1^{i,j,k}; \mathsf{PRF}(\mathsf{k}_1^{i,j,k},))$ for all $k \in [\lambda]$.
     vi. If $i \in I$ then compute $\mathsf{com}_3^i := \mathsf{ExtCom}_3(0^\lambda)$ else compute $\mathsf{com}_3^i := \mathsf{ExtCom}_3(1^\lambda, y, \mathsf{k}_i, \{\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}, \rho^{i,j,k}\}_{j\in[\ell],k\in[\lambda]}; s_i)$.
   (a) Send $\{\mathsf{com}_3^i, \tilde{\Phi}_i\}_{i\in[m]}$ and $\{\mathsf{ct}_0^{i,j,k}, \mathsf{ct}_1^{i,j,k}, \overline{\mathsf{lab}}_{y_{i,j}}^{i,j}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to R.

Lemma 7.7. *Let* ExtCom *be the delayed-input commitment scheme* 1*-rewind secure protocol of Section 2.2. Let* $\mathsf{OT}^{\lambda,m}$ *be a* λ*-out-of-*m *sender private OT (accordingly to Definition 2.6) protocol, then the hybrids* $\mathsf{H}_3^{\alpha-1}$ *and* H_3^α *are computationally indistinguishable.*

Proof. Suppose by contradiction that there exists an adversary that distinguishes the two hybrids with non-negligible probability q, then we show how to use this adversary to contradict the hiding of ExtCom. Let CH be the corresponding challenger for the hiding security game of ExtCom. We make the following two observations before starting the reduction: (1) the simulators $\mathsf{Sim}_1^{1,2}$ and $\mathsf{Sim}_2^{\lambda,m}$ require only a constant number of accepting transcripts in order to perform the extraction of the inputs used by the receiver in the OT executions. Let us denote this constant with R_{ot}; (2) by assumption, the adversary R^* does not abort with non-negligible probability p.

The reduction $\mathcal{A}_{\mathsf{com}}$ works as follows.

1. Upon receiving com_1 from the external challenger, set $\mathsf{com}_1^\alpha \leftarrow \mathsf{com}_1$. Compute the first message accordingly to H_2 (and H_3) with the difference that com_1^α is used.
2. Upon receiving the second round from $R^\star$ act as in H_2 (and H_3), with the following differences:
   (a) Allow $\mathsf{Sim}_1^{1,2}$ and $\mathsf{Sim}_1^{\lambda,m}$ to perform up to R_{ot} rewinds only.
   (b) $\mathsf{Sim}_1^{\lambda,m}$ queries $D'^{\lambda,m}$, instead of $D^{\lambda,m}$. $D'^{\lambda,m}$ behaves exactly like $D^{\lambda,m}$, with the difference that the α-th output (s_α) is set to 0^λ.
   (c) Any time it is required to compute a third round for the α-th execution of the extractable commitment, use in its place a value uniformly random sampled from the space of all the valid third rounds of ExtCom.
3. If R_{ot} accepting transcripts have been collected and it is possible to extract the inputs $\{x_{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}\}$ via $\mathsf{Sim}_1^{1,2}$ and K via $\mathsf{Sim}_1^{\lambda,m}$ then continue as follows, otherwise, output a random bit and stop.
4. If $\alpha \in K$, then output a random bit and stop, else continue.
5. For each $i \in [m], j \in [\ell], k \in [\lambda]$, sample $\mathsf{sk}^{i,j,k} \leftarrow \{0,1\}^\lambda$. Let I be a random subset of K with $|I| = \alpha$, for each $i \in I$, set $s^i := 0^\lambda$.
6. Run $\mathsf{Sim}_2^{1,2}$ and $\mathsf{Sim}_2^{\lambda,m}$ on input respectively $(\{\mathsf{sk}^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}, z_1)$ and $(\{s^i\}_{i\in K}, z_2)$, receive ot_2 and $\{\mathsf{ot}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ from $\mathsf{Sim}_2^{\lambda,m}$ and $\mathsf{Sim}_2^{1,2}$, compute the second round accordingly to $\mathsf{H}_2^{\alpha-1}$ (and H_2^{α}).
7. Upon receiving $\{\mathsf{c},\mathsf{c}'\}$ from $R^\star$ in the second round, parse c as $(\{\mathsf{com}_2^i\}_{i\in[m]}, f)$ and c' as $\{\mathsf{com}_2'^i\}_{i\in[m]}$. and send $(\mathsf{com}_2^\alpha, \mathsf{com}_2'^\alpha)$ to the challenger.
8. Set $m_0 \leftarrow (\mathsf{k}_\alpha, \{\mathsf{sk}_0^{\alpha,j,k}, \mathsf{sk}_1^{\alpha,j,k}, \rho^{\alpha,j,k}\}_{j\in[\ell],k\in[\lambda]})$ and $m_1 \leftarrow 0^\lambda$, and send (m_0, m_1) to the challenger.
9. Upon receiving $(\mathsf{com}_3^\alpha, \mathsf{com}_3'^\alpha)$ from the challenger use these to complete the interaction with $R^\star$ accordingly to $\mathsf{H}_2^{\alpha-1}$ (and H_2^α).

Let q be the non-negligible advantage the adversary has in distinguishing the hybrids $\mathsf{H}_2^{\alpha-1}$ and H_2^α. Let Ext be the event where $\mathsf{Sim}_1^{1,2}$ and $\mathsf{Sim}_1^{\lambda,k}$ extract successfully, then the probability that $\mathcal{A}_{\mathsf{com}}$ wins is $\Pr[\mathsf{Ext}](1/2+q) + (1-\Pr[\mathsf{Ext}])/2 = 1/2 + q\Pr[\mathsf{Ext}]$.

To prove that the reduction is successful we just need to prove the following claim.

Claim 7.8 $\Pr[\mathsf{Ext}]$ *is non-negligible.*

Proof. Suppose that this is not the case, then we can construct a reduction that breaks the sender privacy (Definition 2.6) of $\mathsf{OT}^{\lambda,m}$. The reduction takes as auxiliary input the first round of the adversary, and the inputs encoded in the OT message $\mathsf{ot}_1^{\lambda,m}$ that we denote with K. Note that, due to the sender privacy of $\mathsf{OT}^{\lambda,m}$ we have that $K = \mathsf{OTExt}(\mathsf{ot}_1^{\lambda,m})$. The reduction proceeds as follows.

1. Sample a random $\alpha \in [m]$. For each $i \in [m] \setminus \{\alpha\}$, let s_i be the randomness used in the i-th execution of ExtCom. Send the challenge messages $(K, \{s_1, \ldots, s_\alpha, \ldots, s_m\})$, where $s_\alpha = 0^\lambda$.
2. Upon receiving $\mathsf{ot}^{\lambda,m}$ from the challenger, use it to compute the second round of the protocol.
3. Upon receiving the second round from $R^\star$, complete the third round as in H_2.
4. If $R^\star$ aborts then return 0, else return 1.

We distinguish between two cases, $\alpha \in K$ and $\alpha \notin K$.

When $\alpha \notin K$ then the behavior of $R^\star$ is independent from the input used by the challenger to compute the messages of $\mathsf{OT}^{\lambda,m}$. Indeed, If this is not the case, then we can use the above adversary to break the security of $\mathsf{OT}^{\lambda,m}$.

When $\alpha \in K$, in the worst case, the probability that $R^\star$ provides an accepting third round could be 0.

From the implications above we can conclude that the probability with wich the adversary does not abort (i.e., provide an accepting transcript) is at least $p' = (1 - \lambda/m)p + \lambda/m \cdot 0 = (m - \lambda)p/m$, which is non-negligible. Hence, $\Pr[\,\mathsf{Ext}\,] = p'^{R_{ot}}$. The proof of the claim ends with the observation that R_{ot} is constant.

□

□

Figure 7.7: Hybrid H_4.

Input extraction. Perform the same steps as in H_3^λ.

Forcing the output. Upon receiving $(1^{\kappa_1}, \{x_{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}\}, z_1)$ from $\mathsf{Sim}_1^{1,2}$ and $(1^{\kappa_2}, K, z_2)$ from $\mathsf{Sim}_1^{\lambda,m}$ do the following.

1. Let $\kappa \leftarrow \min\{\kappa_1, \kappa_2\}$, for each $\iota \in [\kappa]$, $i \in [m], j \in [\ell], k \in [\lambda]$, sample $\mathsf{sk}_\iota^{i,j,k} \leftarrow \{0,1\}^\lambda$ if $i \notin K$ then $s_\iota^i := 0^\lambda$ else $s_\iota^i := s^i$.
2. For each $\iota \in [\kappa]$ (i.e., for each rewind performed by the OT simulators $\mathsf{Sim}_2^{1,2}$ and $\mathsf{Sim}_2^{\lambda,m}$ on input respectively $(\{\mathsf{sk}_\iota^{i,j,k}\}_{\iota\in[\kappa],i\in[m],j\in[\ell],k\in[\lambda]}, z_1)$ and $(\{s_\iota^i\}_{\iota\in\kappa,i\in K}, z_2)$, receive ot_2 and $\{\mathsf{ot}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ from $\mathsf{Sim}_2^{\lambda,m}$ and $\mathsf{Sim}_2^{1,2}$ respectively, and do the following.
   (a) Additionally, for each $i \in [m]$ pick $\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k} \leftarrow \{0,1\}^\lambda$ for each $j \in [\ell]$, $k \in [\lambda]$
   (b) For each $i \in K$ compute $\mathsf{ot}_2^{i,j,k} \leftarrow \mathsf{OT}_2^{1,2}(\mathsf{ot}_1^{i,j,k}, (\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}); \rho^{i,j,k})$ for all $j \in [\ell]$ and $k \in [\lambda]$.
   (c) Send ot_2 and $\{\mathsf{ot}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to $R^\star$.
3. Upon receiving two second-round $\{\mathsf{c}, \mathsf{c}'\}$ from $R^\star$, sample $x_S \leftarrow D_S$ and for each $(\{\mathsf{com}_2^i\}_{i\in[m]}, f) \in \{\mathsf{c}, \mathsf{c}'\}$ do the following

```
- For each $i \in [m]$ do the following
  Compute $(y_1, \dots, y_m) \leftarrow \Phi_1(x_S)$.
  If $i \in K$ then
    i. Pick $\mathsf{k}_i \leftarrow \{0,1\}^\lambda$.
    ii. $r_i := \mathsf{PRF}(\mathsf{k}_i, f)$.
    iii. $\tilde{\Phi}_i, \{\overline{\mathsf{lab}}_0^{i,j}, \overline{\mathsf{lab}}_1^{i,j}\}_{i\in[m],j\in[\ell]}, \{\mathsf{lab}_0^{i,j,k}, \mathsf{lab}_1^{i,j,k}\}_{j\in[\ell],k\in[\lambda]} \leftarrow \mathsf{Garble}(1^\lambda, \Phi_{2,i}; r_i)$.
    iv. Sample $\mathsf{k}_0^{i,j,k}, \mathsf{k}_1^{i,j,k} \leftarrow \{0,1\}^*$ and generate $\mathsf{ct}_0^{i,j,k} := \mathsf{Enc}(\mathsf{sk}_0^{i,j,k}, \mathsf{lab}_0^{i,j,k}; \mathsf{PRF}(\mathsf{k}_0^{i,j,k}, ))$,
        $\mathsf{ct}_1^{i,j,k} := \mathsf{Enc}(\mathsf{sk}_1^{i,j,k}, \mathsf{lab}_1^{i,j,k}; \mathsf{PRF}(\mathsf{k}_1^{i,j,k}, ))$ for all $k \in [\lambda]$.
    v. Compute $\mathsf{com}_3^i := \mathsf{ExtCom}_3(1^\lambda, y, \mathsf{k}_i, \{\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}, \rho^{i,j,k}\}_{j\in[\ell],k\in[\lambda]}; s_i)$.
  else
    i. $r_i \leftarrow \{0,1\}^\lambda$.
    ii. $\tilde{\Phi}_i, \{\overline{\mathsf{lab}}_0^{i,j}, \overline{\mathsf{lab}}_1^{i,j}\}_{i\in[m],j\in[\ell]}, \{\mathsf{lab}_0^{i,j,k}, \mathsf{lab}_1^{i,j,k}\}_{j\in[\ell],k\in[\lambda]} \leftarrow \mathsf{Garble}(1^\lambda, \Phi_{2,i}; \underline{r_i})$.
    iii. Sample $r_0^{i,j,k}, r_0^{i,j,k} \leftarrow \{0,1\}^\lambda$ and generate $\mathsf{ct}_0^{i,j,k} := \mathsf{Enc}(\mathsf{sk}_0^{i,j,k}, \mathsf{lab}_0^{i,j,k}; \underline{r_0^{i,j,k}})$, $\mathsf{ct}_1^{i,j,k} :=$
         $\mathsf{Enc}(\mathsf{sk}_1^{i,j,k}, \mathsf{lab}_1^{i,j,k}; \underline{r_1^{i,j,k}})$ for all $k \in [\lambda]$ $j \in [\ell]$.
    iv. $\mathsf{com}_3^i := \mathsf{ExtCom}_3(0^\lambda)$
- Send $\{\mathsf{com}_3^i, \tilde{\Phi}_i\}_{i\in[m]}$ and $\{\mathsf{ct}_0^{i,j,k}, \mathsf{ct}_1^{i,j,k}, \overline{\mathsf{lab}}_{y_{i,j}}^{i,j}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to $R$.
```

Figure 7.8: Hybrid H_5.

Input extraction. Perform the same steps as in H_4.

Forcing the output. Perform step 1 and 2 as in H_4.

```
3. Upon receiving two second-round $\{\mathsf{c}, \mathsf{c}'\}$ from $R^\star$, sample $x_S \leftarrow D_S$ and for each
   $(\{\mathsf{com}_2^i\}_{i\in[m]}, f) \in \{\mathsf{c}, \mathsf{c}'\}$ do the following
   - For each $i \in [m]$ do the following
     Compute $(y_1, \dots, y_m) \leftarrow \Phi(x_S)$.
     If $i \in K$ then
       i. Pick $\mathsf{k}_i \leftarrow \{0,1\}^\lambda$.
       ii. $r_i \coloneqq \mathsf{PRF}(\mathsf{k}_i, f)$.
       iii. $\tilde{\Phi}_i, \{\overline{\mathsf{lab}}_0^{i,j}, \overline{\mathsf{lab}}_1^{i,j}\}_{i\in[m],j\in[\ell]}, \{\mathsf{lab}_0^{i,j,k}, \mathsf{lab}_1^{i,j,k}\}_{j\in[\ell],k\in[\lambda]} \leftarrow \mathsf{Garble}(1^\lambda, \Phi_{2,i}; r_i)$.
       iv. Sample $\mathsf{k}_0^{i,j,k}, \mathsf{k}_1^{i,j,k} \leftarrow \{0,1\}^*$ and generate $\mathsf{ct}_0^{i,j,k} \coloneqq \mathsf{Enc}(\mathsf{sk}_0^{i,j,k}, \mathsf{lab}_0^{i,j,k}; \mathsf{PRF}(\mathsf{k}_0^{i,j,k},))$,
           $\mathsf{ct}_1^{i,j,k} \coloneqq \mathsf{Enc}(\mathsf{sk}_1^{i,j,k}, \mathsf{lab}_1^{i,j,k}; \mathsf{PRF}(\mathsf{k}_1^{i,j,k},))$ for all $k \in [\lambda]$.
       v. Compute $\mathsf{com}_3^i \coloneqq \mathsf{ExtCom}_3(1^\lambda, y, \mathsf{k}_i, \{\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}, \rho^{i,j,k}\}_{j\in[\ell],k\in[\lambda]}; s_i)$.
     else
       i. $r_i \leftarrow \{0,1\}^\lambda$.
       ii. $\tilde{\Phi}_i, \{\overline{\mathsf{lab}}_0^{i,j}, \overline{\mathsf{lab}}_1^{i,j}\}_{i\in[m],j\in[\ell]}, \{\mathsf{lab}_0^{i,j,k}, \mathsf{lab}_1^{i,j,k}\}_{j\in[\ell],k\in[\lambda]} \leftarrow \mathsf{Garble}(1^\lambda, \Phi_{2,i}; r_i)$.
       iii. Generate $\mathsf{ct}_{x^{i,j,k}}^{i,j,k} \coloneqq \mathsf{Enc}(\mathsf{sk}_\iota^{i,j,k}, \mathsf{lab}^{i,j,k}))$, $\underline{\mathsf{ct}_{1-x^{i,j,k}}^{i,j,k} \coloneqq \mathsf{Enc}(\mathsf{sk}_{x^{i,j,k}}^{i,j,k}, 0^\lambda)}$ for all $j \in [\ell]$ and
            $k \in [\lambda]$.
       iv. $\mathsf{com}_3^i \coloneqq \mathsf{ExtCom}_3(0^\lambda)$
   - Send $\{\mathsf{com}_3^i, \tilde{\Phi}_i\}_{i\in[m]}$ and $\{\mathsf{ct}_0^{i,j,k}, \mathsf{ct}_1^{i,j,k}, \overline{\mathsf{lab}}_{y_{i,j}}^{i,j}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to $R$.
```

Figure 7.9: Hybrid H_6.

Input extraction. Perform the same steps as in H_5.
Forcing the output. Perform step 1 and 2 as in H_5.

```
3. Upon receiving two second-round $\{\mathsf{c}, \mathsf{c}'\}$ from $R^\star$, sample $x_S \leftarrow D_S$, compute $(y_1, \dots, y_m) \leftarrow$
   $\Phi_1(x_S)$. For each $(\{\mathsf{com}_2^i\}_{i\in[m]}, f) \in \{\mathsf{c}, \mathsf{c}'\}$ do the following
   - For each $i \in [m]$ do the following
     If $i \in K$ then
       i. Pick $\mathsf{k}_i \leftarrow \{0,1\}^\lambda$.
       ii. $r_i \coloneqq \mathsf{PRF}(\mathsf{k}_i, f)$.
       iii. $\tilde{\Phi}_i, \{\overline{\mathsf{lab}}_0^{i,j}, \overline{\mathsf{lab}}_1^{i,j}\}_{i\in[m],j\in[\ell]}, \{\mathsf{lab}_0^{i,j,k}, \mathsf{lab}_1^{i,j,k}\}_{j\in[\ell],k\in[\lambda]} \leftarrow \mathsf{Garble}(1^\lambda, \Phi_{2,i}; r_i)$.
       iv. Sample $\mathsf{k}_0^{i,j,k}, \mathsf{k}_1^{i,j,k} \leftarrow \{0,1\}^*$ and generate $\mathsf{ct}_0^{i,j,k} \coloneqq \mathsf{Enc}(\mathsf{sk}_0^{i,j,k}, \mathsf{lab}_0^{i,j,k}; \mathsf{PRF}(\mathsf{k}_0^{i,j,k},))$,
           $\mathsf{ct}_1^{i,j,k} \coloneqq \mathsf{Enc}(\mathsf{sk}_1^{i,j,k}, \mathsf{lab}_1^{i,j,k}; \mathsf{PRF}(\mathsf{k}_1^{i,j,k},))$ for all $k \in [\lambda]$.
       v. Compute $\mathsf{com}_3^i \coloneqq \mathsf{ExtCom}_3(1^\lambda, y, \mathsf{k}_i, \{\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}, \rho^{i,j,k}\}_{j\in[\ell],k\in[\lambda]}; s_i)$.
     else
       i. $\underline{\text{Compute } z_i \leftarrow \Phi_i(\{x_{i,j,k}\}_{j\in[\ell],k\in[m]}, \{y_{i,j}\}_{j\in[\ell]})}$
       ii. $\underline{\tilde{\Phi}_i, \{\overline{\mathsf{lab}}^{i,j}\}_{j\in[\ell]}, \{\mathsf{lab}^{i,j,k}\}_{j\in[\ell],k\in[\lambda]} \leftarrow \mathsf{Sim}_{\mathsf{GC}}(1^\lambda, 1^{|\Phi_i|}, 1^{2\ell}, z_i).}$
       iii. Generate $\mathsf{ct}_{x^{i,j,k}}^{i,j,k} \coloneqq \mathsf{Enc}(\mathsf{sk}_\iota^{i,j,k}, \mathsf{lab}^{i,j,k}))$, $\mathsf{ct}_{1-x^{i,j,k}}^{i,j,k} \coloneqq \mathsf{Enc}(\mathsf{sk}_{x^{i,j,k}}^{i,j,k}, 0^\lambda)$ for all $j \in [\ell]$ and
            $k \in [\lambda]$.
       iv. $\mathsf{com}_3^i \coloneqq \mathsf{ExtCom}_3(0^\lambda)$
   - Send $\{\mathsf{com}_3^i, \tilde{\Phi}_i\}_{i\in[m]}$ and $\{\mathsf{ct}_0^{i,j,k}, \mathsf{ct}_1^{i,j,k}, \overline{\mathsf{lab}}_{y_{i,j}}^{i,j}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to $R$.
```

Figure 7.10: H_7.

Input extraction. Perform the same steps as in H_6.
Forcing the output. Perform step 1 and 2 as in H_6.

3. Upon receiving two second-round $\{\mathsf{c}, \mathsf{c}'\}$ from $R^\star$ in the ι-th rewind, sample $x_S \leftarrow D_S$ and for each $(\{\mathsf{com}_2^i\}_{i\in[m]}, f) \in \{\mathsf{c}, \mathsf{c}'\}$ do the following
   (a) $\underline{\text{Run } \mathsf{Sim}_\Phi \text{ by corrupting the client corresponding to the receiver and the set of servers indexed by } K \text{ and obtain } \{y_i\}_{i\in K}.}$
   (b) $\underline{\text{For each } i \in [m], j \in [\ell] \text{ compute } x_{i,j} = \oplus_{k\in[\lambda]} x_{i,j,k}. \text{ Let } x_i = x_{i,1}||\ldots||x_{i,\ell}. \text{ Run } \mathsf{Sim}_\Phi \text{ on input } \{x_i\}_{i\notin K} \text{ as the first round message sent by the malicious client and } f \text{ as the function to be computed. When } \mathsf{Sim}_\Phi \text{ makes a query to the ideal functionality on input } x_R, \text{ intercept this query and compute } \mathsf{out} := f(x_S, x_R). \text{ Provide } \mathsf{out} \text{ as the output from the ideal functionality to } \mathsf{Sim}_\Phi \text{ and obtain } \{z_i\}_{i\notin K}.}$
   (c) For each $i \in [m]$ do the following
   (d) If $i \in K$ then
      i. Pick $\mathsf{k}_i \leftarrow \{0,1\}^\lambda$.
      ii. $r_i := \mathsf{PRF}(\mathsf{k}_i, f)$.
      iii. $\tilde{\Phi}_i, \{\overline{\mathsf{lab}}_0^{i,j}, \overline{\mathsf{lab}}_1^{i,j}\}_{i\in[m],j\in[\ell]}, \{\mathsf{lab}_0^{i,j,k}, \mathsf{lab}_1^{i,j,k}\}_{j\in[\ell],k\in[\lambda]} \leftarrow \mathsf{Garble}(1^\lambda, \Phi_{2,i}; r_i)$.
      iv. Sample $\mathsf{k}_0^{i,j,k}, \mathsf{k}_1^{i,j,k} \leftarrow \{0,1\}^*$ and generate $\mathsf{ct}_0^{i,j,k} := \mathsf{Enc}(\mathsf{sk}_0^{i,j,k}, \mathsf{lab}_0^{i,j,k}; \mathsf{PRF}(\mathsf{k}_0^{i,j,k},)),$ $\mathsf{ct}_1^{i,j,k} := \mathsf{Enc}(\mathsf{sk}_1^{i,j,k}, \mathsf{lab}_1^{i,j,k}; \mathsf{PRF}(\mathsf{k}_1^{i,j,k},))$ for all $k \in [\lambda]$.
      v. Compute $\mathsf{com}_3^i := \mathsf{ExtCom}_3(1^\lambda, y, \mathsf{k}_i, \{\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}, \rho^{i,j,k}\}_{j\in[\ell],k\in[\lambda]}; s_i)$.

      else
      i. $\tilde{\Phi}_i, \{\overline{\mathsf{lab}}^{i,j}\}_{j\in[\ell]}, \{\mathsf{lab}^{i,j,k}\}_{j\in[\ell],k\in[\lambda]} \leftarrow \mathsf{Sim}_{\mathsf{GC}}(1^\lambda, 1^{|\Phi_i|}, 1^{2\ell}, z_i)$.
      ii. Generate $\mathsf{ct}_{x^{i,j,k}}^{i,j,k} := \mathsf{Enc}(\mathsf{sk}_\iota^{i,j,k}, \mathsf{lab}^{i,j,k})), \mathsf{ct}_{1-x^{i,j,k}}^{i,j,k} := \mathsf{Enc}(\mathsf{sk}_{x^{i,j,k}}^{i,j,k}, 0^\lambda)$ for all $j \in [\ell]$ and $k \in [\lambda]$.
      iii. $\mathsf{com}_3^i := \mathsf{ExtCom}_3(0^\lambda)$
   - Send $\{\mathsf{com}_3^i, \tilde{\Phi}_i\}_{i\in[m]}$ and $\{\mathsf{ct}_0^{i,j,k}, \mathsf{ct}_1^{i,j,k}, \overline{\mathsf{lab}}_{y_{i,j}}^{i,j}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to R.

Figure 7.11: Simulator Sim_R against malicious receivers.

Sim_1. Run $\mathsf{Sim}_1^{1,2}$ and $\mathsf{Sim}_1^{\lambda,m}$. For each rewind performed by the simulators prior to the extraction do the following
1. Let $\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}$ be the input sampled by $\mathsf{Sim}_1^{1,2}$ from $D^{1,2}$, for each $i \in [m], j \in [\ell], k \in [\lambda]$, and let $s_1, \ldots, s_m$ the values that $\mathsf{Sim}_1^{\lambda,m}$ samples when querying $D^{\lambda,m}$.
2. Compute $\mathsf{com}_1^i := \mathsf{ExtCom}_1(1^\lambda; s_i)$ for all $i \in [m]$ and send $\{\mathsf{com}_1^i\}_{i\in[m]}$ to $R^\star$.
3. Upon receiving ot_1^i and $\{\mathsf{ot}_1^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ from $R^\star$ forward these to $\mathsf{Sim}_1^{\lambda,m}$ and $\mathsf{Sim}_1^{1,2}$ respectively.
4. When $\mathsf{Sim}_1^{1,2}$ or $\mathsf{Sim}_1^{\lambda,m}$ send $\{\mathsf{ot}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ and ot_2 respectively, forward these messages to $R^\star$ along with x_S with $x_S \leftarrow D_S$.
5. Upon receiving two second-round $\{\mathsf{c}, \mathsf{c}'\}$ from $R^\star$, for each $(\{\mathsf{com}_2^i\}_{i\in[m]}, f) \in \{\mathsf{c}, \mathsf{c}'\}$, where f_0 is the function received in c and f_1 is the function received in c' do the following
   - For each $i \in [m]$ do the following
      i. Sample $\mathsf{k}_i \leftarrow \{0,1\}^\lambda$ for all $i \in [m]$.
      ii. Compute $(y_1, \ldots, y_m) \leftarrow \Phi_1(x_S)$.
      iii. Compute $\mathsf{com}_3^i := \mathsf{ExtCom}_3(1^\lambda, y_i, \mathsf{k}_i, \{\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}, \rho^{i,j,k}\}_{j\in[\ell],k\in[\lambda]}; s_i)$ for all $i \in [m]$ (where $\rho^{i,j,k}$ is the randomness used by $\mathsf{Sim}_1^{1,2}$ to compute the message $\mathsf{ot}_2^{i,j,k}$).
      iv. Compute $\tilde{\Phi}_i, \{\overline{\mathsf{lab}}_0^{i,j}, \overline{\mathsf{lab}}_1^{i,j}\}_{i\in[m],j\in[\ell]}, \{\mathsf{lab}_0^{i,j,k}, \mathsf{lab}_1^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]} \leftarrow \mathsf{Garble}(1^\lambda, \Phi_{2,i}, r_i)$ for all $i \in [m]$ where $r_i := \mathsf{PRF}(\mathsf{k}_i, f)$. Here $\{\overline{\mathsf{lab}}_0^{i,j}, \overline{\mathsf{lab}}_1^{i,j}\}_{i\in[m],j\in[\ell]}$ are the input labels for the sender and $\{\mathsf{lab}_0^{i,j,k}, \mathsf{lab}_1^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ are the input labels for the receiver.

   v. Generate $\mathsf{ct}_b^{i,j,k} := \mathsf{Enc}(\mathsf{sk}_b^{i,j,k}, \mathsf{lab}_b^{i,j,k}; \mathsf{PRF}(\mathsf{k}_b^{i,j,k}, f))$ for all $b \in \{0,1\}, i \in [m], j \in [\ell]$ and $k \in [\lambda]$.

(a) Send $\tilde{C}, \{\mathsf{ct}_0^i, \mathsf{ct}_1^i, \mathsf{com}_3^i, \tilde{\Phi}_i\}_{i\in[m]}$ and $\{\mathsf{ct}_0^{i,j,k}, \mathsf{ct}_1^{i,j,k}, \overline{\mathsf{lab}}_{y_{i,j}}^{i,j}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to $R^\star$.

6. Upon receiving ot_3 and $\{\mathsf{ot}_3^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ forward these to $\mathsf{Sim}_1^{\lambda,m}$ and $\mathsf{Sim}_1^{1,2}$ respectively.
7. Upon receiving $(1^{\kappa_1}, \{x_{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}\}, z_1)$ from $\mathsf{Sim}_1^{1,2}$ and $(1^{\kappa_2}, K, z_2)$ from $\mathsf{Sim}_1^{\lambda,m}$, for each $i \in [m], j \in [\ell]$ compute $x_{i,j} = \oplus_{k\in[\lambda]} x_{i,j,k}$. Let $x_i = x_{i,1}||\ldots||x_{i,\ell}$, and reconstruct the input of the corrupted receiver x_R.
8. Let $\kappa \leftarrow \min\{\kappa_1, \kappa_2\}$, return $(1^\kappa, (x_R, f_0, f_1), \mathsf{aux} = (K, \{x_{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}\}, z_1, z_2)$.
9. For each $\iota \in [\kappa]$, $i \in [m], j \in [\ell], k \in [\lambda]$, sample $\mathsf{sk}_\iota^{i,j,k} \leftarrow \{0,1\}^\lambda$ if $i \notin K$ then $s_\iota^i := 0^\lambda$, else $s_\iota^i := s^i$.
10. For each $\iota \in [\kappa]$ (i.e., for each rewind performed by the OT simulators $\mathsf{Sim}_2^{1,2}$ and $\mathsf{Sim}_2^{\lambda,m}$ on input respectively $(\{\mathsf{sk}_\iota^{i,j,k}\}_{\iota\in[\kappa],i\in[m],j\in[\ell],k\in[\lambda]}, z_1)$ and $(\{s_\iota^i\}_{\iota\in\kappa,i\in K}, z_2)$, receive ot_2 and $\{\mathsf{ot}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ from $\mathsf{Sim}_2^{\lambda,m}$ and $\mathsf{Sim}_2^{1,2}$ respectively, and do the following.
    (a) For each $i \in [m]$ pick $\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k} \leftarrow \{0,1\}^\lambda$ for each $j \in [\ell]$, $k \in [\lambda]$
    (b) For each $i \in K$ compute $\mathsf{ot}_2^{i,j,k} \leftarrow \mathsf{OT}_2^{1,2}(\mathsf{ot}_1^{i,j,k}, (\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}); \rho^{i,j,k})$ for all $j \in [\ell]$ and $k \in [\lambda]$.
    (c) Sample $x_S^\iota \leftarrow \mathcal{X}^S$.
    (d) Send ot_2 and $\{\mathsf{ot}_2^{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to $R^\star$ along with x_S^ι.
11. Upon receiving two second-round $\{\mathsf{c}, \mathsf{c}'\}$ from $R^\star$ in the ι-th rewind, and for each $(\{\mathsf{com}_2^i\}_{i\in[m]}, f) \in \{\mathsf{c}, \mathsf{c}'\}$ do the following
    (a) Run Sim_Φ by corrupting the client corresponding to the receiver and the set of servers indexed by K and obtain $\{y_i\}_{i\in K}$. Run Sim_Φ on input $\{x_i\}_{i\notin K}$ as the first round message sent by the malicious client and f as the function to be computed. When Sim_Φ makes a query to the ideal functionality on input x_R, intercept this query provide $\mathsf{out}^\iota \leftarrow f(x_S^\iota, x_R)$ as the output from the ideal functionality to Sim_Φ and obtain $\{z_i\}_{i\notin K}$.
    (b) For each $i \in [m]$ do the following
    (c) If $i \in K$ then
       i. Pick $\mathsf{k}_i \leftarrow \{0,1\}^\lambda$.
       ii. $r_i := \mathsf{PRF}(\mathsf{k}_i, f)$.
       iii. $\tilde{\Phi}_i, \{\overline{\mathsf{lab}}_0^{i,j}, \overline{\mathsf{lab}}_1^{i,j}\}_{i\in[m],j\in[\ell]}, \{\mathsf{lab}_0^{i,j,k}, \mathsf{lab}_1^{i,j,k}\}_{j\in[\ell],k\in[\lambda]} \leftarrow \mathsf{Garble}(1^\lambda, \Phi_{2,i}; r_i)$.
       iv. Sample $\mathsf{k}_0^{i,j,k}, \mathsf{k}_1^{i,j,k} \leftarrow \{0,1\}^*$ and generate $\mathsf{ct}_0^{i,j,k} := \mathsf{Enc}(\mathsf{sk}_0^{i,j,k}, \mathsf{lab}_0^{i,j,k}; \mathsf{PRF}(\mathsf{k}_0^{i,j,k},))$, $\mathsf{ct}_1^{i,j,k} := \mathsf{Enc}(\mathsf{sk}_1^{i,j,k}, \mathsf{lab}_1^{i,j,k}; \mathsf{PRF}(\mathsf{k}_1^{i,j,k},))$ for all $k \in [\lambda]$.
       v. Compute $\mathsf{com}_3^i := \mathsf{ExtCom}_3(1^\lambda, y, \mathsf{k}_i, \{\mathsf{sk}_0^{i,j,k}, \mathsf{sk}_1^{i,j,k}, \rho^{i,j,k}\}_{j\in[\ell],k\in[\lambda]}; s_i)$.
    else
       i. $\tilde{\Phi}_i, \{\overline{\mathsf{lab}}^{i,j}\}_{j\in[\ell]}, \{\mathsf{lab}^{i,j,k}\}_{j\in[\ell],k\in[\lambda]} \leftarrow \mathsf{Sim}_{\mathsf{GC}}(1^\lambda, 1^{|\Phi_i|}, 1^{2\ell}, z_i)$.
       ii. Generate $\mathsf{ct}_{x^{i,j,k}}^{i,j,k} := \mathsf{Enc}(\mathsf{sk}_\iota^{i,j,k}, \mathsf{lab}^{i,j,k}))$, $\mathsf{ct}_{1-x^{i,j,k}}^{i,j,k} := \mathsf{Enc}(\mathsf{sk}_{x^{i,j,k}}^{i,j,k}, 0^\lambda)$ for all $j \in [\ell]$ and $k \in [\lambda]$.
       iii. $\mathsf{com}_3^i := \mathsf{ExtCom}_3(0^\lambda)$
    - Send $\{\mathsf{com}_3^i, \tilde{\Phi}_i\}_{i\in[m]}$ and $\{\mathsf{ct}_0^{i,j,k}, \mathsf{ct}_1^{i,j,k}, \overline{\mathsf{lab}}_{y_{i,j}}^{i,j}\}_{i\in[m],j\in[\ell],k\in[\lambda]}$ to R.
12. Upon receiving $\delta^{1,2}$ from $\mathsf{Sim}_2^{1,2}$ and $\delta^{\lambda,m}$ from $\mathsf{Sim}_2^{\lambda,m}$. If $\delta^{1,2} = \delta^{\lambda,m}$, set $\delta := \delta^{1,2}$,and continue, else abort the execution.
13. Save the state of Sim_1 aux, and return $(1^\kappa, (x_R, f_0, f_1, z)$, where (f_0, f_1) represent the function sent from the receiver in the δ-th thread.

Sim_2. Upon receiving $(\{\mathsf{out}_0^\iota, \mathsf{out}_1^\iota\}_{\iota\in\kappa}, \mathsf{aux} = (K, \{x_{i,j,k}\}_{i\in[m],j\in[\ell],k\in[\lambda]}\})$ continue the δ-th thread from the third round, and do all the steps that Sim_1 does from point $11.a$ until point (excluded) point 13 up to κ times, with the only difference that when Sim_Φ makes a query to the ideal functionality on input x_R, intercept this query and reply with out^ι (when Sim_Φ is queried the ι-th time for the ideal functionality f_b) as the output from the ideal functionality to Sim_Φ and obtain $\{z_i\}_{i\notin K}$.

Additional property against adversarial receivers. Suppose that the function the receiver picks is always of the following form:

- Let $x_R := (x, \beta)$ be the input of the receiver, and $x_S := (y, s_0, s_1)$ be the input of the sender
- The function $f^{\alpha,g}$ returns $g(x, y), s_{\beta \oplus \alpha}$, where $\beta, \alpha \in \{0, 1\}$, and g is just another NC_1 function.

Consider the simulator $\tilde{\mathsf{Sim}}_1$ that works exactly like Sim_1 with the following differences

- During the input extraction phase, instead of sampling from $\mathcal{X}_S$, Sim_1 samples from $\mathcal{X}^d_S$. $\mathcal{X}^d_S$ queries $\mathcal{X}_S$ thus obtaining (y, s_0, s_1) and returns (y, s_d, s_d).
- Sim_1 rewinds the adversarial receiver only R^{ot} times (i.e., $\mathsf{Sim}^{1,2}_1$ and $\mathsf{Sim}^{\lambda,n}_1$ stop after R^{ot} rewinds).

Assuming that we have an adversary that provides a valid third round with non-negligible probability ε, then $\tilde{\mathsf{Sim}}_1$ extracts correctly the input of the adversarial receiver with non-negligible probability either when $d = 0$ or when $d = 1$. More formally, let Ext^d be the event where $\tilde{\mathsf{Sim}}_1$ extracts correctly the input of the adversarial receiver when querying $\mathcal{X}^d_S$, then we have that

Claim 7.9 $\Pr\left[\exists d \in \{0, 1\} | \mathsf{Ext}^d\right]$ *is non-negligible.*

Proof. If by contradiction the claim does not hold it means that $\tilde{\mathsf{Sim}}_1$ fails when $d = 0$ and $d = 1$. In particular, this means that the probability that $\tilde{\mathsf{Sim}}_1$ collects R^{ot} accepting transcripts is negligible in both cases, hence, this means that the aborting probability of the adversary changes by a non-negligible amount, compared with when the adversarial receiver is interacting with Sim_1. Note that in the threads where $d = \alpha \oplus \beta$ the view of the adversary in Sim_1 is identical to the view of the adversary in $\tilde{\mathsf{Sim}}_1$. To conclude the proof we need to argue that $d = \alpha \oplus \beta$ with non-negligible probability in at least R^{ot} rewinding threads. Suppose that this probability is negligible both when $d = 0$ and when $d = 1$, then this would contradict what we just argued (i.e., it would contradict the security of Sim_1). □

Theorem 7.10. *Let $\ell = \mathrm{poly}(\lambda)$ and $\{X_{S_j}\}_{j\in[\ell]}$ be a high min-entropy random variable defined by a probability distribution $\{\mathcal{X}_{S_j}\}_{j\in[\ell]}$, the protocol Π^{2PC} (Figure 7.1) is 1-rewinding secure ℓ-sender list-simulatable against sometimes aborting adversaries with delayed function selection for NC_1 circuits for the ideal functionality $\mathcal{F}^{\{X_{S_j}\}_{j\in[\ell]}}$ (according to Definition 3.4), that makes block box use of $\mathsf{OT}^{1,2}, \mathsf{OT}^{k,\lambda}, \mathsf{ExtCom}, \mathsf{PRF}, \mathsf{SE}$, and GC.*

The proof of this theorem proceeds very similar to Theorem 7.1. In particular, the proof differs slightly only in proving the security against ℓ corrupted receivers. In this case, we observe that the proof uses the same hybrid arguments $\{\mathsf{H}_i\}_{i\in[7]}$ of Section 7.2, but each hybrid H_i is iterated ℓ times one for each execution of Π^{2PC} which is performed between the j-th sender and the j-th receiver for $j \in [\ell]$. The indistinguishability between the hybrids is argued similarly to the proof of Section 7.2.

8 List Non-Malleable OT

8.1 Definition ℓ-Non-Malleable k-out-of-m Oblivious Transfer

In this section, we give the definition of list non-malleable k-out-of-m Oblivious Transfer. Roughly speaking, we consider an adversarial man-in-the-middle (denoted by MIM) that interacts with up to ℓ senders $S_1, \ldots, S_\ell$ on the left, and up to ℓ receivers $R_1, \ldots, R_\ell$ on the right. The guarantee of non-malleability ensures that the distribution of the inputs that the adversary uses in the right sessions (as a sender) is independent of whether the adversary receives messages generated from honest senders, or generated from a simulator (who uses only the inputs that muset appear in the output view of the adversarial receiver). More formally, This is formalized by the existence of a simulator S which interacts with MIM as senders in the left sessions and as honest receivers in the right sessions. S having access to the list-simulatable ideal OT functionality $\mathcal{F}^{\mathsf{list}}_{\mathsf{NMOT}}$ (Figure 8.2), is able to simulate $S_1, \ldots, S_\ell$.

This definition is very close in spirit to the one described in [IKSS21], but adapted to our list simulatability paradigm.

Definition 8.1 (ℓ-Non-Malleable k-out-of-m Oblivious Transfer). *An ℓ-non-malleable Oblivious Transfer protocol ($\mathsf{NM\text{-}OT}$) is a protocol between a sender S with inputs $\{x_i\}_{i\in[m]}$ from a the domain $\mathcal{X}$ and a receiver R with input $K \subset [m]$ where $|K| = k$, that satisfies the following properties:*

Correctness: *For every $i \in [m], x_i \in \{0,1\}^\lambda$ and $K \subset [m]$ such that $|K| = k$,*

$$\mathsf{out}_R\langle S(\{x_i\}_{i\in[m]}, R(K))\rangle = \{x_i\}_{i\in K}$$

Receiver Security (under Parallel Composition with Fixed Roles):
For every non-uniform PPT adversary S^* there exists a non-uniform PPT adversary Sim for the ideal world such that

$$\mathrm{Real}_{\mathsf{NM\text{-}OT},S^*}(1^\lambda)\} \approx \mathrm{Ideal}_{F^m_{\mathsf{OT}},\mathsf{Sim}}(1^\lambda)$$

where $\mathrm{Real}_{\mathsf{NM\text{-}OT},S^*}$ denotes the distribution of the output of the adversary S^* (controlling the sender) after a real execution of the protocol $\mathsf{NM\text{-}OT}$ where the receiver has inputs $\{K_j\}_{j\in[\ell]}$ and the sender has input $\{x_i\}_{i\in[m]}$. $\mathrm{Ideal}_{F^m_{\mathsf{OT}},\mathsf{Sim}}$ denotes the analogous distribution in an ideal execution with a trusted party that computes F^m_{OT} (see Figure 8.1) for the parties and hands the output to the receiver.

Figure 8.1: Functionality F^m_{OT}

***Functionality** F^m_{OT}*

F^m_{OT} running with a sender S, a receiver R and an adversary Sim proceeds as follows:

- *Upon receiving a message $(\mathsf{send}, \{x_i\}_{i\in[m]}, S, R)$ from S where each $x_i \in \{0,1\}^\lambda$, record $\{x_i\}_{i\in[m]}$ and send send to R and Sim. Ignore any subsequent send messages.*
- *Upon receiving a message $(\mathsf{receive}, K)$ from R, where $K \subset [m]$ send $\{x_c\}_{c\in K}$ to R and $\mathsf{receive}$ to S and Sim and halt. (If no $(\mathsf{send}, \cdot)$ message was previously sent, do nothing).*

List Non-Malleability: *Consider any PPT adversary (denoted by MIM), with any auxiliary input aux, that interacts with up to ℓ senders $S_1, \ldots, S_\ell$ on a* left session *and up to ℓ receivers $R_1, \ldots, R_\ell$ on a* right, *where for every $j \in [\ell], R_j$ has input K_j. Let $\mathsf{Real}_{\mathsf{MIM},\mathsf{NM\text{-}OT}}(1^\lambda, \{K_j\}_{j\in[\ell]})$ denote the joint distribution of the view of the adversary and the outputs of honest parties in the real-world world experiment described in Figure 8.3. Moreover let $\mathsf{inp}_{\mathsf{MIM}} = \{\tilde{x}_{i,j}\}_{i\in[M],j\in H}$ be the inputs of MIM implicitly defined in the real-world experiment while MIM is acting as a malicious sender. We require the existence of an expected non-uniform polynomial time simulator Sim that with black-box access to the adversary MIM interacts with the ideal functionality $\mathcal{F}^{\mathsf{list}}_{\mathsf{NMOT}}$ (see Figure 8.2) as described in the ideal world experiment of Figure 8.3, and return an output, denoted by $\mathsf{Ideal}_{\mathsf{Sim},\mathcal{F}^{\mathsf{list}}_{\mathsf{NMOT}}}(1^\lambda, \{K_j\}_{j\in[\ell]})$ such that the following holds for any $\{K_j\}_{j\in[\ell]}$*

$$\mathsf{Real}_{\mathsf{MIM},\mathsf{NM\text{-}OT}}(1^\lambda, \{K_j\}_{j\in[\ell]}), \mathsf{inp}_{\mathsf{MIM}} \approx_c \mathsf{Ideal}_{\mathsf{Sim},\mathcal{F}^{\mathsf{list}}_{\mathsf{NMOT}}}(1^\lambda, \{K_j\}_{j\in[\ell]})$$

Figure 8.2: The list functionality $\mathcal{F}^{\mathsf{list}}_{\mathsf{NMOT}}$

The functionality is parametrized by the samplers $\mathcal{X}$, and by the honest receivers inputs $\{K_j\}_{j\in[\ell]}$.

1. *Upon receiving the receivers-inputs of the corrupted parties $\{\tilde{K}_j\}_{j\in[\ell]}$, and $k \in \mathbb{N}$, for each $\kappa \in [k]$ do the following*
   - *For each $j \in [\ell]$ sample $\{x_{i,j,\kappa}\}_{i\in[m]} \leftarrow \mathcal{X}$ and compute $\mathsf{out}^\kappa_{i,j} := \{x_{c,j,\kappa}\}_{c\in\tilde{K}_j}$*
2. *Send $\{\mathsf{out}^\kappa_j\}_{\kappa\in[k],j\in[\ell]}$ to MIM.*
3. *Upon receiving (abort) from MIM do the following. For each $j \in \mathsf{abort}$ send an abort command to P_j.*

Figure 8.3: Real and ideal world

$\mathsf{Real}_{\mathsf{MIM},\mathsf{NM\text{-}OT}}(1^\lambda, \{K_j\}_{j\in[\ell]})$

1. *For each* $i \in [m], j \in [\ell]$ *sample* $x_{i,j} \leftarrow \mathcal{X}^i$.
2. *For each* $j \in [\ell]$ *compute the first round* $\pi^1_{S_j}$ *of* NM-OT *as the honest sender* S_j *would do on input* $(\{x_{i,j}\}_{i\in[m]})$ *and the randomness* $r_j \leftarrow \{0,1\}^\lambda$ *and send* $\pi^1_{S_j}$ *to* MIM.
3. *For each* $j \in [\ell]$ *compute the first round* $\pi^1_{R_j}$ *of* NM-OT *as the honest receiver* R_j *would do on input* (K_j) *and the randomness* $r_j \leftarrow \{0,1\}^\lambda$ *and send* $\pi^1_{R_j}$ *to* MIM.
4. *For each* $q \in \{2,3\}$
   (a) *For each* $j \in [\ell]$ *upon receiving* $\tilde{\pi}^{q-1}_{R_j}$ *from* MIM *(playing as a malicious receiver), compute the* q*-th round* $\tilde{\pi}^q_{S_j}$ *of* NM-OT *as the honest sender* S_j *would do and send it to* MIM.
   (b) *For each* $j \in [\ell]$ *upon receiving* $\tilde{\pi}^{q-1}_{S_j}$ *from* MIM *(playing as a malicious sender), compute the* q*-th round* $\tilde{\pi}^q_{R_j}$ *of* NM-OT *as the honest receiver* R_j *would do and send it to* MIM.
5. *For each* $j \in [\ell]$ *upon receiving* $\tilde{\pi}^3_{R_j}$, *compute the output* out_j *as* S_j *would do, and return the view of* MIM.

$\mathsf{Ideal}_{\mathsf{Sim},\mathcal{F}^{\mathsf{list}}_{\mathsf{NMOT}}}(1^\lambda, \{K_j\}_{j\in[\ell]})$

1. $(\{\tilde{K}_j\}_{j\in[\ell]}, 1^k, z) \leftarrow \mathsf{Sim}^{\mathcal{A}}_1(1^\lambda, \{K_j\}_{j\in[\ell]})$
2. *Send* $\{\tilde{K}_j\}_{j\in[\ell]}$ *to the ideal functionality* $\mathcal{F}^{\mathsf{list}}_{\mathsf{NMOT}}$ *which computes* $\{\mathsf{out}^\kappa_j\}_{j\in[\ell],\kappa\in[k]}$ *as described using the sampled inputs* $\{x_{i,j}\}_{i\in[m],j\in[\ell]}$ *as described before.*
3. *Whenever* $\mathsf{Sim}^{\mathcal{A}}_2(\{\mathsf{out}^\kappa_j\}_{j\in[\ell],\kappa\in[k]}, z)$ *queries* $\mathcal{A}$ *with a message* $(\{\pi^q_{S_j}\}_{j\in[\ell],q\in[3]})$, *replace the query with* $(\{\pi^q_{S_j}\}_{j\in[\ell],q\in[3]}, \{x_{i,j}\}_{i\in[m],j\in[\ell]})$ *and forward the pair to* $\mathcal{A}$.
4. $(\iota, \text{View}, \{\tilde{x}_{i,j}\}_{i\in[M],j\in H}) \leftarrow \mathsf{Sim}^{\mathcal{A}}_2(\{\mathsf{out}^\kappa_j\}_{j\in[\ell],\kappa\in[k]}, z)$ *with* $\iota \in [k] \cup \{\perp\}$.
5. *Send* (abort) *where* abort *is a set of indices that determines which honest parties should not receive the output from* $\mathcal{F}^{\mathsf{list}}_{\mathsf{NMOT}}$.
6. *Return* View, $\{\tilde{x}_{i,j}\}_{i\in[M],j\in H}$.

8.2 Construction of ℓ-Non-Malleable k-out-of-m OT

In this section, we gave our construction of ℓ non-malleable list-simulatable k-out-of-m OT (formally described in Figure 8.5).

Our construction is similar to the one presented in [IKSS21], the crucial difference is that we use our three-round two-party computation protocol Π, which satisfies the notion of list simulatability against sometimes aborting (instead of a simulation secure two-party computation protocol). To construct our 3-round non-malleable OT protocol we make use of the following tools.

– An information-theoretic $m(\lambda)\cdot\ell(\lambda)$ non-malleable coding scheme/split-state non-malleable code $\mathsf{NM} = (\mathsf{Code}, \mathsf{Decode})$.
– A low-depth proof $\mathsf{ldp} = (\mathsf{Prove}, \mathsf{Verify})$ for P.
– An existentially unforgeable signature scheme $\mathsf{DS} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$.
– A 3-round public-coin syncronous non-malleable commitment with simulatability property $\Pi_{\mathsf{NMC}} = (S_{\mathsf{NMC}}, R_{\mathsf{NMC}})$.
– A three-round list simulatable, against sometimes aborting adversaries, two-party computation protocol Π accordingly to Definition 3.4 for NC^1 circuits (constructed in Section 7). Π implements the function $f_{\mathsf{NM-OT}}$ described in Figure 8.4, the corresponding ideal list functionality parametrized by the distribution $\mathcal{D}_{\mathsf{NM-OT}}$ defined over the sender's inputs, i.e. the distribution induced by the digital signature scheme, the non-malleable commitment scheme, the non-malleable code, as well as the uniform distribution. More formally, this distribution is defined by the distributions derived from $\mathsf{DS.Setup}$, $2\cdot\lambda\cdot m$ random values, $\lambda\cdot m$ codewords generated by $\mathsf{NM.Code}$ as well as $2\cdot\lambda\cdot m$ commitments and decommitments generated by the execution of the commitment phase of Π_{NMC}.

We propose the formal description of our k-out-of-m NM-OT protocol in Figure 8.5 and refer the reader to the introductory section for an informal discussion on how the protocol works.

Figure 8.4: Function $f_{\mathsf{NM\text{-}OT}}$ parametrized with $(c_1, \ldots, c_\lambda)$

$\underline{f_{\mathsf{NM\text{-}OT}}.}$

S's input: (x, ldp) with $x = (\mathsf{vk}, \{(s'_{L,i,j}, s'_{R,i,j}), (L_{i,j}, R_{i,j}, x_j), (\mathsf{dec}_{L,i,j}, s_{L,i,j}), (\mathsf{dec}_{R,i,j}, s_{R,i,j})\}_{i\in[\lambda],j\in[m]})$. R's input: indices K and $\{\mathsf{k}_i\}_{i\in[\lambda]}$.

The function $f_{\mathsf{NM-OT}}$ upon receiving the inputs defined above does the following.

1. Set $x' = (\mathsf{vk}, \{(s'_{L,i,j}, s_{L,i,j}, s'_{R,i,j}, s_{R,i,j}), (L_{i,j}, R_{i,j}, x_j)\}_{i\in[\lambda],j\in[m]})$.
2. If $\mathsf{LDP.Verify}(x', \mathsf{ldp}) \neq 1$, output $\perp$.
3. Otherwise set $\mathsf{out} = (\mathsf{vk}, \{x_j\}_{j\in K})$. Additionally, for every $i \in [\lambda]$, if $c_i \oplus \mathsf{k}_i = 0$, append $\{(\mathsf{dec}_{L,i,j}, s_{L,i,j})\}_{j\in[m]}$ to out and if $c_i \oplus \mathsf{k}_i = 1$, append $\{(\mathsf{dec}_{R,i,j}, s_{R,i,j})\}_{j\in[m]}$ to out.
4. Append $\{x_j\}_{j\in K}$ and $\{s'_{R,i,j}, s'_{L,i,j}\}_{i\in\lambda,j\in[m]}$ to out.
5. Output out.

Figure 8.5: Our $\ell(\lambda)$ Non-Malleable k-out-of-m OT protocol.

Initialization: The sender S has inputs $\{x_j\}_{j\in[m]}$ and the receiver R uses as its input a set $K \subseteq [m]$ where $|K| = k$
Protocol: S and R interact in the following way:

1. S generates $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{DS.Setup}(1^\lambda)$ and does the following:
   - For each $i \in [\lambda], j \in [m]$, sample $r_{i,j} \leftarrow \{0,1\}^*$ and compute $(L_{i,j}, R_{i,j}) := \mathsf{NM.Code}((\mathsf{vk}\|x_j); r_{i,j})$.
   - For each $i \in [\lambda], j \in [m]$:
     - Sample $s_{L,i,j}$ and $s_{R,i,j} \leftarrow \{0,1\}^*$.
     - Engage with R in a commitment phase of Π_{NMC} w.r.t message $s_{L,i,j}$, producing commitment transcript $\mathsf{com}_{L,i,j}$ and decommitment $\mathsf{dec}_{L,i,j}$.
     - Engage with R in a commitment phase of Π_{NMC} w.r.t message $s_{R,i,j}$, producing commitment transcript $\mathsf{com}_{R,i,j}$ and decommitment $\mathsf{dec}_{R,i,j}$[a].
     - Set $s'_{L,i,j} = L_{i,j} \oplus s_{L,i,j}$ and $s'_{R,i,j} = R_{i,j} \oplus s_{R,i,j}$.
   - Set $x' = (\mathsf{vk}, \{(s'_{L,i,j}, s_{L,i,j}, s'_{R,i,j}, s_{R,i,j}), (L_{i,j}, R_{i,j}, x_j)\}_{i\in[\lambda],j\in[m]})$ and $\mathcal{L} = \{(\mathsf{vk}, \{(s'_{L,i,j}, s_{L,i,j}, s'_{R,i,j}, s_{R,i,j}), (L_{i,j}, R_{i,j}, x_j)\}_{i\in[\lambda],j\in[m]}) : \forall i \in [\lambda], j \in [m], L_{i,j} = s'_{L,i,j} \oplus s_{L,i,j}, R_{i,j} = s'_{R,i,j} \oplus s_{R,i,j}, \mathsf{NM.Decode}(L_{i,j}, R_{i,j}) = (\mathsf{vk}\|x_j)\}$. Then compute $\mathsf{ldp} = \mathsf{LDP.Prove}(x', \mathcal{L})$.
2. Both parties engage in a run of the two-party protocol Π where:
   - R acts as the receiver using as its private input K and $(\mathsf{k}_1, \ldots, \mathsf{k}_\lambda)$, and parametrizing the delayed function with $(c_1, \ldots, c_\lambda)$, where $\mathsf{k}_i, c_i \leftarrow \{0,1\}$ for all $i \in [\lambda]$.
   - S acts as the sender using (x, ldp) as its input where $x = (\mathsf{vk}, \{(s'_{L,i,j}, s'_{R,i,j}), (L_{i,j}, R_{i,j}, x_j), (\mathsf{dec}_{L,i,j}, s_{L,i,j}), (\mathsf{dec}_{R,i,j}, s_{R,i,j})\}_{i\in[\lambda],j\in[m]})$.
   Additionally, S signs the messages generated from Π, denoted by (Π_1, Π_2, Π_3). It sets $\sigma_i \leftarrow \mathsf{DS.Sign}(\mathsf{sk}, \Pi_i)$ for all $i \in [3]$ and sends it to R.
3. R parses $\mathsf{out} = (\mathsf{vk}, \{(\mathsf{dec}_{i,j}, s_{i,j})\}_{j\in[m]}, \{x_j\}_{j\in K}, \{s'_{L,i,j}, s'_{R,i,j}\}_{i\in[\lambda],j\in[m]})$. It outputs $\{x_j\}_{j\in K}$ if and only if the following conditions hold:
   (a) $\mathsf{DS.Verify}(\mathsf{vk}, \Pi_i, \sigma_i)$ for all $i \in [3]$.
   (b) For each $i \in [\lambda]$ compute $c_i \oplus k_i = \gamma_i$. If $\gamma_i = 0$ then $\{(\mathsf{dec}_{i,j}, s_{i,j})\}_{j\in[m]}$ (contained in out) are valid decommitment information for $\{\mathsf{com}_{L,i,j}\}_{i\in[\lambda],j\in[m]}$, else $\{(\mathsf{dec}_{i,j}, s_{i,j})\}_{j\in[m]}$ are valid decommitment information for $\{\mathsf{com}_{R,i,j}\}_{i\in[\lambda],j\in[m]}$.

[a] The commitment phases are executed in parallel with the execution of Π.

Theorem 8.2. *Let $\mathcal{X}$ be a high min-entropy random variable defined by a probability distribution $\mathcal{D}$, then the k-out-of-m OT* NM-OT *described in Figure 8.5 satisfies Definition 8.1 for the functionality $\mathcal{F}_{\mathsf{OT}}^{\mathsf{list},\mathcal{X}}$, against sometimes aborting adversaries. Moreover* NM-OT *makes black-box use of* Π, Π_{com}, NMC *and* DS.

Proof. The correctness of the described protocol follows directly from the correctness of the underlying primitives.

List non-malleability. We prove now that the protocol in Figure 8.5 satisfies list non-malleability.

This proof proceeds by hybrid arguments. In the first hybrid, we start by considering a real-world execution where the MIM interacts with ℓ honest senders on the left and ℓ honest receivers on the right. The last hybrid corresponds to the experiments where the MIM interacts with the simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$.

Hybrid H_0: This hybrid corresponds to the real world. In more detail, in this hybrid, we have an execution of the MIM with ℓ honest senders $\{S_l\}_{l\in[\ell]}$ on the left, using inputs $\{x_{j,l} \in \mathcal{X}\}_{j\in[m]}$, and ℓ honest receivers on the right, using indices $(\{K_l\}_{l\in[\ell]})$.

Hybrid H_1: This hybrid proceeds as the previous one but the hybrid extracts the inputs $\{\tilde{x}_{j,l}\}_{j\in[m],l\in[\ell]}$ played by MIM as malicious sender. The hybrid proceeds with the extraction in the following way. Let $p(\lambda)$ the probability that the MIM completes this execution without aborting and set $\gamma(\lambda) = \max(\lambda, p^{-2}(\lambda))$. The hybrid does a first honest execution (let us indicate it as the main execution). Then if this execution is accepting for an honest receiver the hybrid proceeds by fixing the first round received in the right session (in the first execution) and rewinding the MIM from the 3rd to the 2nd round $\gamma(\lambda)$ times. In each rewind, the hybrid (acting as honest receiver) takes the values $(c^l_1, \dots, c^l_\lambda)$ uniformly at random (note that these values can be specified in the second round as they are part of the function computed by Π), for each of the ℓ executions (i.e. $l \in [\ell]$). If the MIM completes again in one of this rewind, let $(c'^l_1, \dots, c'^l_\lambda)$ be the value chosen (acting as honest receiver) by the hybrid. For every $l \in [\ell]$, let index $\alpha \in [\lambda]$ be such that $c^l_\alpha \oplus \mathsf{k}^l_\alpha = 0$ and $c'^l_\alpha \oplus \mathsf{k}^l_\alpha = 1$. Moreover, let $\widetilde{s'}_{L^l,\alpha,j}, \widetilde{s'}_{R^l,\alpha,j}$ for each $j \in [m]$ be the value obtained in the main execution. The hybrid uses the values $\widetilde{\mathsf{dec}}_{L^l,\alpha,j}, \tilde{s}_{L^l,\alpha,j}, \widetilde{\mathsf{dec}}_{R^l,\alpha,j}, \tilde{s}_{R^l,\alpha_j,j}$ (obtained as output from the main and rewinding executions) and the values $\widetilde{s'}_{L^l,\alpha,j}, \widetilde{s'}_{R^l,\alpha,j}$ to compute $\tilde{L}^l_{\alpha,j}, \tilde{R}^l_{\alpha,j}$ and subsequently $\tilde{x}_{j,l}$.
If no such rewinding thread exists, or if there exists $l \in [\ell]$ for which there does not exist $\alpha \in [\lambda]$ such that $c^l_\alpha \oplus \mathsf{k}^l_\alpha = 0$ and $c'^l_\alpha \oplus \mathsf{k}^l_\alpha = 1$, then set $\tilde{x}_{j,l} = \perp$ for all $i \in [m]$.
The output of this hybrid is the joint distribution

$$\mathrm{View}_{\mathsf{MIM}}\langle\{S^l(\{x_{j,l}\}_{j\in[m]})\}_{l\in[\ell]}, \{R^l(K_l)\}_{j\in[\ell]}\rangle, \{\tilde{x}_{j,l}\}_{i\in[m],l\in[\ell]}$$

Hybrid H_2: This hybrid proceeds as the previous hybrid, with the difference that, instead of the honest protocol, the simulator $\Pi.\mathsf{Sim}_S$ of Π is executed in all sessions where the MIM acts as a receiver. The hybrid in all the ℓ sessions where she was following the honest sender procedure using input $\{x_{j,l}\}_{j\in[m],l\in[\ell]}$ she stars, instead, executing $\Pi.\mathsf{Sim}_S = (\Pi.\mathsf{Sim}^1_S, \Pi.\mathsf{Sim}^2_S)$ parametrized by the distribution $\mathcal{D}_{\mathsf{NM-OT}}$ Specifically, the hybrids run $\Pi.\mathsf{Sim}^1_S$ acting as ℓ malicious receivers, and it does so acting as a proxy for the messages of Π between $\Pi.\mathsf{Sim}^1_S$ and MIM in all the sessions where the MIM acts as a receiver. Upon receiving $(\{\tilde{K}_l, \{k_{i,l}\}_{i\in[m]}, f^l_0, f^l_1\}_{l\in[\ell]}, 1^\kappa)$ from $\mathsf{NM\text{-}OT.Sim}^1$, the hybrid acts as the ideal functionality of Π would do and computes output $\{\mathsf{out}^b_{j,\kappa}\}_{j\in[\ell],\kappa\in[k],b\in\{0,1\}}$ using honest sender's input sampled from $\mathcal{D}_{\mathsf{NM-OT}}$. At this point the hybrid runs $\Pi.\mathsf{Sim}^2_S$ on input $\{\mathsf{out}^b_{j,\kappa}\}_{j\in[\ell],\kappa\in[k],b\in\{0,1\}}$ and acts with her as the ideal functionality of Π would do (using the honest sender's inputs previously sampled). Moreover, the hybrid acts as a proxy for the messages of Π between $\Pi.\mathsf{Sim}^2_S$ and MIM in all the sessions where the MIM acts as a receiver. Finally the hybrid receives $(\iota, \mathrm{View}_{\mathsf{MIM}})$ from $\Pi.\mathsf{Sim}^2_S$.
Receiver messages for the right executions are generated exactly as in H_1 w.r.t. honest receiver input K_l. The extraction of the values $\{\tilde{x}_{j,l}\}_{j\in[m],l\in[\ell]}$ proceeds exactly as in H_1 with the only differences that also in the rewinding threads the messages of the honest sender are simulated. The output of this hybrid is $\mathrm{View}_{\mathsf{MIM}}$ and $\{\tilde{x}_{j,l}\}_{j\in[m],l\in[\ell]}$.

Hybrid H_3: This hybrid proceeds as the previous hybrid, but defining inp_{S_l} differently, for all $l \in [\ell]$. inp_{S_l} is defined as described in H_2 except for the values $\{s'_{L^l,i,j}$ and $s'_{R^l,i,j}\}_{i\in[\lambda]}$, which are described as specified below. Let $(\mathsf{k}^l_1, \dots, \mathsf{k}^l_\lambda)$ be the values that $\Pi.\mathsf{Sim}_S$ queries to the ideal functionality for the l-th session. Moreover, let $(c^l_1, \dots, c^l_\lambda)$ be the values that MIM sent as input of the delayed functions for the l-th session. Then, the hybrid answers (acting as the ideal functionality) to $\Pi.\mathsf{Sim}_S$ using the honest sender's inputs (i.e., using inp_{S_l} as described in H_2) except that if $\tilde{c}^l_i \oplus \mathsf{k}^l_i = 0$ the value $s'_{R^l,i,j}$ is set as a random string, and otherwise the value $s'_{L^l,i,j}$ is set as a random string, for each $i \in [\lambda]$.

Finally, note that in the rewinding threads, the values $(\tilde{c}^l_1, \dots, \tilde{c}^l_\lambda)$ that MIM sent as input of the delayed functions change, and the values $\{s'_{L^l,i,j}$ and $s'_{R^l,i,j}\}_{i\in[\lambda]}$ are adjusted consequently. It is important to notice that after this hybrid the answers given by the hybrid (acting as the ideal functionality) in a thread are dependent only on one of the two states of the non-malleable code.

Hybrid H_4: This hybrid proceeds as the previous hybrid, with the difference that if the challenger obtains a verification key in one of the right sessions that is identical to a verification key used in one of the left sessions, this hybrid outputs $\perp$. The indistinguishability between this and the previous hybrid follows from the unforgeability of the digital signature scheme DS.

Hybrid H_5: This hybrid corresponds to the ideal world. It proceeds as the previous hybrid with the difference that inp_{S_l} is differently defined. In more detail, for every $l \in [\ell], i \in [m]$ and $\alpha \in [\lambda]$, the hybrid sets $(L^j_{\alpha,i}, R^j_{\alpha,i}) \leftarrow \mathsf{NM.Sim}(1^{p(\lambda)})$.
We note that at this point, the functionality $\mathcal{F}^{\mathsf{list}}_{OT^{\alpha,\beta}}$ can be perfectly simulated with access to the ideal functionality $\mathcal{F}^{\mathsf{list}}_{\mathsf{NMOT}}$. Therefore, the output of this hybrid is identical to the ideal view.
The indistinguishability between this and the previous hybrid follows from the security of the non-malleable codes NM.

Lemma 8.3. *The hybrids H_0 and H_1 are indistinguishable.*

Proof. We start the proof by making the following two claims:

1. After the MIM executes the first round the values $\widetilde{\mathsf{dec}}_{L^l,\alpha,j}, \tilde{s}_{L^l,\alpha,j}, \widetilde{\mathsf{dec}}_{R^l,\alpha,j}, \tilde{s}_{R^l,\alpha_j,j}$ are fixed due to the statistically binding property of Π_{NMC}.
2. Due to the soundness of Idp and the security of Π it is possible to argue that the values $\tilde{L}^l_{\alpha,j}, \tilde{R}^l_{\alpha,j}$ are correctly reconstructed from $\tilde{s}_{L^l,\alpha,j} \oplus \tilde{s}'_{L^l,\alpha,j}$ and $\tilde{s}_{R^l,\alpha,j} \oplus \tilde{s}'_{R^l,\alpha,j}$.

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

Lemma 8.4. *Let Π be the 1-rewinding ℓ-senders secure two-party computation protocol constructed in Section 7, then the hybrids H_1 and H_2 are indistinguishable.*

Proof. Assuming an adversary MIM that distinguishes between the hybrids H_1 and H_2 with probability $q(\lambda)$ then we construct an ℓ adversarial receivers $R^*_1, \dots, R^*_\ell$ that breaks the the 1-rewinding ℓ-senders security of the underlying Π protocol. Let $\mathcal{C}$ be the corresponding challenger.

The reduction construct and adversary $R^*_1, \dots, R^*_\ell$ which runs internally MIM acting as described both in hybrids H_1 and H_2, i.e. acting as an honest receivers on the right executions and as an honest senders on the left executions where the reduction is acting as a proxy between $\mathcal{C}$ and MIM. To be more precise, the reduction computes the following steps:

1. For each $l^* \in [\ell]$ the reduction simulates the l^*-th malicious receiver $R^*_{l^*}$ for $\mathcal{C}$ in the following way.
   (a) Obtain the first round message for the left execution externally from $\mathcal{C}$, and use it to generate the first round of the protocol accordingly to H_1 and H_2, for the the l^*'th left execution. Moreover in the right executions generate the first round messages according to the receiver strategy using the inputs $\{K_j\}_{j\in[\ell]}$. Obtain first-round messages from the MIM (both on the left and the right executions), and output the MIM's message in the l^*'th left execution to $\mathcal{C}$.
   (b) Obtain the second round message for the left execution externally from $\mathcal{C}$, and use it to generate the second round of the protocol accordingly to H_1 and H_2, for the l^*'th left execution.
   (c) In the right executions generate the second round messages according to the honest receiver strategy.
   (d) Obtain second-round messages from the MIM (both on the left and the right executions),
   (e) Rewind MIM one-time and repeat steps (c) and (d). Then, output the MIM's message in the l^*'th left execution to $\mathcal{C}$ (for both of the two threads).

   (f) Obtain the (two) third round messages for the left execution externally from $\mathcal{C}$, and use it to generate the third round of the protocol accordingly to H_1 and H_2, for the l^*'th left execution (for both of the two threads).
   (g) In the right executions (for both of the two threads) generate the third-round messages according to the honest receiver strategy.
   (h) For both threads obtain third-round messages from the MIM (both on the left and the right executions), and output the MIM's message in the l^*'th left execution to $\mathcal{C}$.
   (i) If none of the two executions abort computes the values $\{\tilde{x}_{j,l}\}_{j\in[m],l\in[\ell]}$ as describe in both H_1 and H_2.
2. The reduction output the view of $R_1^*, \ldots, R_\ell^*$ in the above experiments together with $\{\tilde{x}_{j,l}\}_{j\in[m],l\in[\ell]}$.

Note that (1) If $\mathcal{C}$ sends simulated messages the experiments is distributed as H_1 and H_2 otherwise; (2) by our assumption MIM is non aborting with some non-negligible probability $p(\lambda)$ in H_1, moreover notice that by the security of Π follows that the probability with which MIM aborts in H_2 is a probability $p_*(\lambda)$ negligibly close to $p(\lambda)$. Therefore the reduction has non-negligible probability $p_*^2(\lambda) \cdot q(\lambda)$ to succeed. □

Lemma 8.5. *Let Π_{NMC} be a 3-round syncronous public coin non-malleable commitment with simulatability property then the hybrids H_3 and H_4 are computationally indistinguishable.*

Proof. For the proof of this lemma, we consider the intermediate hybrids $\mathsf{H}^l_{0,j}, \mathsf{H}^l_{1,j}, \ldots, \mathsf{H}^l_{\lambda,j}$ where for every $l \in [\ell]$, $j \in [m]$ and $i \in [\lambda]$, hybrid $\mathsf{H}^l_{i,j}$, is identical to hybrid $\mathsf{H}^l_{i-1,j}$, except for that one of the value $\{s'_{L^l,i,j}$ and $s'_{R^l,i,j}\}_{i\in[\lambda]}$ is chosen as a random string. More specifically, if $c^l_{i,j} \oplus \mathsf{k}^l_{i,j} = 0$ the value $s'_{R^l,i,j}$ is set as a random string, and otherwise the value $s'_{L^l,i,j}$ is set as a random string, for each $i \in [\lambda]$, where $c^l_{i,j}$ is the i-th input in the j-th session for the delayed function and $\mathsf{k}^l_{i,j}$ is the i-th input played by MIM acting as a receiver in the j-th session.

To prove this lemma, we now prove the indistinguishability of the hybrids $\mathsf{H}^l_{i-1,j}$, and $\mathsf{H}^l_{i,j}$, for all $i \in [\lambda]$ by contradiction. In more detail, assuming there exists a distinguisher $\mathcal{D}_{\mathsf{H}}$ which distinguishes between the hybrids $\mathsf{H}^l_{i^*-1,j}$, and $\mathsf{H}^l_{i^*,j}$ with probability $q(\lambda)$, for a $i^* \in [\lambda]$, then we construct an adversarial man-in-the-middle MIM^* that breaks the non-malleability of the underlying Π_{NMC} protocol. Let $\mathcal{C}$ be the challenger for this game.

The reduction constructs and adversary MIM^* which has i^* as advice. On the start of the experiment MIM^* sets the values $s'_{R^l,i^*,j}$, $s_{R^l,i^*,j}$ and $s'_{L^l,i^*,j}$, $s_{L^l,i^*,j}$ as described in $\mathsf{H}_{1,i^*-1,i}$. Then, MIM^* flips a bit b at random and chooses a string r at random. If $b = 0$ MIM^* sends to $\mathcal{C}$ the values $(s_{L^l,i^*,j}, r)$ and the values $(s_{R^l,i^*,j}, r)$ otherwise.

MIM^* runs now the MIM acting as described both in $\mathsf{H}^l_{i^*-1,j}$, and $(\mathsf{H}^l_{i^*,j})$, i.e. acting as an honest receiver on the right executions and as an honest sender in the left executions, except that in l-th session. In this execution, the reduction acts as a proxy for the messages of Π_{NMC} and $\mathcal{C}$ acting as a sender in the left execution and as a receiver in the right execution.

In particular, the reduction MIM^* acts as follows.

1. Upon receiving com^1 from the external challenger as a first round of the commitment phase of Π_{NMC}, set $\mathsf{com}^1_{L^l,i^*,j} \leftarrow \mathsf{com}^1$ if $b = 0$ and as $\mathsf{com}^1_{R^l,i^*,j} \leftarrow \mathsf{com}^1$ otherwise. Compute the rest of the first message of the left session, running $\Pi.\mathsf{Sim}^1_S$, accordingly to $\mathsf{H}^l_{i^*-1,j}$, and $(\mathsf{H}^l_{i^*,j})$.
2. Upon receiving the first round from MIM in the right session forward the first round of Π_{NMC} to the external challenger, namely forward $\widetilde{\mathsf{com}}^1_{L^l,i^*,j}$ if $b = 0$ and $\widetilde{\mathsf{com}}^1_{R^l,i^*,j}$ otherwise. Let $\widetilde{\mathsf{com}}^2$ the second round of Π_{NMC} obtained from the external challenger, set $\widetilde{\mathsf{com}}^2_{L^l,i^*,j} \leftarrow \mathsf{com}^2$ if $b = 0$ and as $\mathsf{com}^2_{R^l,i^*,j} \leftarrow \mathsf{com}^2$ otherwise. Compute the rest of the second message of the right session accordingly to $\mathsf{H}^l_{i^*-1,j}$, and $(\mathsf{H}^l_{i^*,j})$.
3. Compute the second message of the left session, running $\Pi.\mathsf{Sim}^1_S$, accordingly to $\mathsf{H}^l_{i^*-1,j}$, and $(\mathsf{H}^l_{i^*,j})$.
4. Upon receiving the second round (and specifically the value $c^l_{i^*,j}$) from MIM in the right session act as in $\mathsf{H}^l_{i^*-1,j}$, and $(\mathsf{H}^l_{i^*,j})$, with the following differences:
   (a) Allow $\Pi.\mathsf{Sim}^1_S$ to perform up to R_Π rewinds only.

(b) $\Pi.\mathsf{Sim}^1_S$ queries $\mathcal{D}'_{\mathsf{NM-OT}}$, instead of $\mathcal{D}_{\mathsf{NM-OT}}$. $\mathcal{D}'_{\mathsf{NM-OT}}$ behaves exactly like $\mathcal{D}_{\mathsf{NM-OT}}$, with the following difference: if $b = 0$ the decommitment information of $\mathsf{com}_{L^l,i^*,j}$ are set to 0^λ, otherwise the decommitment information of $\mathsf{com}_{R^l,i^*,j}$ are set to 0^λ.

(c) Any time it is required to compute a second round for non-malleable commitment obtained from the challenger in the right session, use in its place a value uniformly random sampled from the uniform distribution (note that this is possible since Π_{NMC} is public-coin).

(d) Any time it is required to compute a third round for non-malleable commitment obtained from the challenger in the left session, use in its place a value uniformly random sampled from the space of all the valid third rounds of Π_{NMC} (note that this is possible since Π_{NMC} enjoys the simulatability property).

5. If R_Π accepting transcripts have been collected and it is possible to extract the inputs of MIM acting as a receiver in the left session and specifically the value then continue as follows, otherwise, output a random bit and stop.
6. If $\mathsf{k}^l_{i^*,j} \oplus c^l_{i^*,j} = b$, then output a random bit and stop, else continue.
7. Compute the second message on the left session accordingly to $\mathsf{H}^l_{i^*-1,j}$, and $(\mathsf{H}^l_{i^*,j})$, running $\Pi.\mathsf{Sim}^2_S$ on input the honest seder's input as specified in $\mathsf{H}^l_{i^*-1,j}$, with difference that if if $b = 0$ the decommitment information of $\mathsf{com}_{L^l,i^*,j}$ are set to 0^λ, otherwise the decommitment information of $\mathsf{com}_{L^l,i^*,j}$ are set to 0^λ.
8. Compute the second message of the right session accordingly to $\mathsf{H}^l_{i^*-1,j}$, and $(\mathsf{H}^l_{i^*,j})$. In particular use as honest input of the receiver in the right session the strings $(\tilde{\mathsf{k}}^l_1, \dots, \tilde{\mathsf{k}}^l_\lambda)$, $(\tilde{c}^l_1, \dots, \tilde{c}^l_\lambda)$, where $\tilde{\mathsf{k}}^l_{i^*,j} \oplus \tilde{c}^l_{i^*,j} = 1 - b$ and all the rest are chosen from the uniform distribution over $\{0, 1\}$.
9. Upon receiving the 2nd round in the left session from MIM forwards the second round of Π_{NMC} to the external challenger obtaining the 3rd round of the commitment phase of Π_{NMC}, namely com^3.
Set $\mathsf{com}^3_{L^l,i^*,j} \leftarrow \mathsf{com}^3$ if $b = 0$ and as $\mathsf{com}^3_{R^l,i^*,j} \leftarrow \mathsf{com}^3$ otherwise. Compute the rest of the third message of the left session, running $\Pi.\mathsf{Sim}^2_S$, accordingly to $\mathsf{H}^l_{i^*-1,j}$, and $(\mathsf{H}^l_{i^*,j})$.
10. Compute the third message of the right session accordingly to $\mathsf{H}^l_{i^*-1,j}$, and $(\mathsf{H}^l_{i^*,j})$.
11. Upon receiving the third round from MIM in the right session reconstruct from MIM the value $\tilde{R}^l_{i^*,j}$ if $b = 0$ and the value $\tilde{L}^l_{i^*,j}$ if $b = 1$, moreover forward the third round of Π_{NMC} to the external challenger.
12. Generate a new second-round message according to the honest receiver strategy on the right executions, and proceed as described in steps from (4) to (10) described above.
13. If none of the two executions abort computes the values $\{\tilde{x}_{j,l}\}_{j\in[m],l\in[\ell]}$ as describe in both $\mathsf{H}^l_{i^*-1,j}$, and $(\mathsf{H}^l_{i^*,j})$.
14. Output the view of MIM^* and the values $\{\tilde{x}_{j,l}\}_{j\in[m],l\in[\ell]}$ as well as the value $\tilde{L}^l_{i^*,j}$ if $b = 1$ (respectively the value $\tilde{R}^l_{i^*,j}$ if $b = 0$).

We now describe the distinguisher $\mathcal{D}_{\mathsf{NMC}}$ for Π_{NMC} which, by definition, has on input the view of MIM^* and the message committed in the right session by MIM^* which is $\tilde{L}^l_{i^*,j}$ if $b = 0$ (respectively the value $\tilde{R}^l_{i^*,j}$ if $b = 1$). The values $\{\tilde{x}_{j,l}\}_{j\in[m],l\in[\ell]}$ and the value $\tilde{L}^l_{i^*,j}$ if $b = 1$ (respectively the value $\tilde{R}^l_{i^*,j}$ if $b = 0$) are set as an auxiliary input. $\mathcal{D}_{\mathsf{NMC}}$ sets the value $\tilde{x}_{j,l}$ as $\tilde{R}^l_{i^*,j} \oplus \tilde{L}^l_{i^*,j}$, while the remaining $\{\tilde{x}_{j,l}\}_{j\in[m],l\in[\ell]}$are left untouched. At this point $\mathcal{D}_{\mathsf{NMC}}$ runs $\mathcal{D}_{\mathsf{H}}$ on input $\{\tilde{x}_{j,l}\}_{j\in[m],l\in[\ell]}$ and the view of MIM^* given in output the output of $\mathcal{D}_{\mathsf{H}}$.

Note that (1) If $\mathcal{C}$ sends a commitment of r the experiment is distributed as $\mathsf{H}^l_{i^*,j}$ and as $\mathsf{H}^l_{i^*-1,j}$ otherwise. (2) The extraction of $\tilde{\mathsf{k}}^l_{i^*,j}$ (conditioned on $\tilde{\mathsf{k}}^l_{i^*,j} \oplus \tilde{c}^l_{i^*,j} \neq b$) is successful with non-negligible probability after a constant number of rewinds given Claim 7.9. (3) MIM is not aborting in the 3rd round of the hybrid $\mathsf{H}^l_{i^*-1,j}$ with probability $p(\lambda)$, she is aborting in the 3rd round of the hybrid $\mathsf{H}^l_{i^*,j}$ with probability negligibly close to $p(\lambda)$, due to the hiding of Π_{NMC}. (4) Pre and post-rewind the distribution of MIM inputs acting as a receiver in the left sessions does not change due to the hiding of Π_{NMC}.

Therefore, we can conclude that the reduction is successful with non-negligible probability. □

Lemma 8.6. *Let* NM *be* $m(\lambda)\cdot\ell(\lambda)$ *symmetric non-malleable codes satisfying Definition 2.7, then the hybrids* H_5 *and* H_4 *are indistinguishable against an unbounded adversary.*

Proof. For the proof of this lemma, we consider the intermediate hybrids $\{\mathsf{H}^l_{4,i,j}\}_{j\in[m],l\in[\ell],i\in[\lambda]\cup\{0\}}$ where:

1. $\mathsf{H}_4 = \mathsf{H}^{\ell}_{4,\lambda,0}, \mathsf{H}_5 = \mathsf{H}^{\ell}_{4,\lambda,m}$,
2. for $j \in [m], \mathsf{H}^{\ell}_{4,\lambda,j-1} = \mathsf{H}^1_{4,0,j}$,
3. for $l \in [\ell], \mathsf{H}^{l-1}_{4,\lambda,j} = \mathsf{H}^l_{4,0,j}$,
4. for every $j \in [m], l \in [\ell], i \in [\lambda], \mathsf{H}^l_{4,i,j}$ is identical to $\mathsf{H}^l_{4,i-1,j}$ except that $\mathsf{H}^l_{4,i,j}$ samples $(L^l_{i,j}, R^l_{i,j}) \leftarrow$ $\mathsf{NM.Code}(0)$.

To prove this lemma, we now prove the indistinguishability of the hybrids $\mathsf{H}^l_{4,i-1,j}$ and $\mathsf{H}^l_{4,i,j}$ for all $j \in [m], l \in [\ell], i \in [\lambda]$ by contradiction. Then there exist i^*, j^*, l^* and an abounded distinguisher that distinguish hybrids $\mathsf{H}^{l^*}_{4,i^*-1,j^*}$ and $\mathsf{H}^{l^*}_{4,i^*,j^*}$

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

1. $f_{\mathsf{MIM}}, g_{\mathsf{MIM}}$ and h_{MIM} share a common state that is generated as follows:
   (a) Execute $\Pi.\mathsf{Sim}$ using the honest receiver strategy in the right executions with input $\{K_l\}_{l\in[\ell]}$ and uniformly chosen $\{c^l_1, \ldots, c^l_\lambda\}_{l\in[\ell]}, \{\mathsf{k}^l_1, \ldots, \mathsf{k}^l_\lambda\}_{j\in[\ell]}$ until $\Pi.\mathsf{Sim}$ generates a query to the ideal functionality at the end of the second round.
   (b) At this point, $\Pi.\mathsf{Sim}$ outputs a view and transcript of the MIM until the second round as well as $\{\tilde{K}_l, \tilde{c}^l_1, \ldots, \tilde{c}^l_\lambda\}_{l\in[\ell]}$ and $\{\tilde{\mathsf{k}}^l_1, \ldots, \tilde{\mathsf{k}}^l_\lambda\}_{l\in[\ell]}$ that corresponds to the receiver's input in the left executions.
   (c) Rewind the third round once with uniformly and independently chosen $\{c'^l_1, \ldots, c'^l_\lambda\}_{l\in[\ell]}$.
   (d) Obtain the rewinding view (with the same first round) and obtain MIM's input which is acting as a receiver in these sessions $\{\bar{c}^l_1, \ldots, \bar{c}^l_\lambda\}_{l\in[\ell]}$ (note that the rest of the MIM's inputs, acting as receivers, are fixed after the first round). If $\tilde{c}^l_i \oplus \tilde{\mathsf{k}}^l_i \neq \bar{c}^l_i \oplus \tilde{\mathsf{k}}^l_i$, continue. Otherwise, abort.
   (e) Generate $(L^l_{i,j}, R^l_{i,j})$ for every $(j,l,i) \in [m]\times[\ell]\times[\lambda] \setminus \{j^*, l^*, i^*\}$ according to $\mathsf{H}^{l^*}_{4,i^*-1,j^*}$ (respectively $\mathsf{H}^{l^*}_{4,i^*,j^*}$).
   (f) Output the view of the MIM until round 2 in the main rewinding threads, including (i^*, j^*, l^*), the values $(L^l_{i,j}, R^l_{i,j})_{(j,l,i)\in[m]\times[\ell]\times[\lambda]\setminus\{i^*,j^*,k^*\}}$.
   (g) Additionally, output the receiver's inputs $(\{\tilde{K}_l, \tilde{c}^l_1, \ldots, \tilde{c}^l_\lambda, \tilde{\mathsf{k}}^l_1, \ldots, \tilde{\mathsf{k}}^l_\lambda\}_{l\in[\ell]})$ and the sender's inputs $\{(\mathsf{vk}^l, x^l_j, \mathsf{com}_{L^l,i,j}, \mathsf{com}_{R^l,i,j}, (\mathsf{dec}_{L^l,i,j}, s_{L^l,i,j}), (\mathsf{dec}_{R^l,i,j}, s_{R^l,i,j})\}_{i\in[\lambda],j\in[m],l\in[\ell]}$.
2. Next, the function h_{MIM} on input L, sets $L^{l^*}_{i^*,j^*} = L, R^{l^*}_{i^*,j^*} = 0$. The function has hardwired values $(\{\tilde{K}_l, \tilde{c}^l_1, \ldots, \tilde{c}^l_\lambda, \tilde{\mathsf{k}}^l_1, \ldots, \tilde{\mathsf{k}}^l_\lambda\}_{l\in[\ell]})\{(\mathsf{vk}^l, x^l_j, \mathsf{com}_{L^l,i,j}, \mathsf{com}_{R^l,i,j}, (\mathsf{dec}_{L^l,i,j}, s_{L^l,i,j}), (\mathsf{dec}_{R^l,i,j}, s_{R^l,i,j})\}_{i\in[\lambda],j\in[m],l\in[\ell]}$ as well as the values $(L^l_{i,j}, R^l_{i,j}, s'_{L^l,i,j}, s'_{R^l,i,j})_{(j,l,i)\in[m]\times[\ell]\times[\lambda]\setminus\{i^*,j^*,k^*\}}$. The function sets the value $s'_{L^{l^*},i^*,j^*} = s_{L^{l^*},i^*,j^*} \oplus L$ and $s'_{R^{l^*},i^*,j^*}$ as a random string. The function using the hardwired values and $s'_{L^{l^*},i^*,j^*}, s'_{R^{l^*},i^*,j^*}$ computes the output out for the honest senders in the all ℓ left executions. It then invokes $\Pi.\mathsf{Sim}^2$ on out to generate the third round message of the protocol transcript in the main thread if $\tilde{c}^{l^*}_{i^*} \oplus \tilde{\mathsf{k}}^{l^*}_{i^*} = 0$, and generates the third round message of the protocol transcript in the rewinding thread if $\bar{c}^{l^*}_{i^*} \oplus \tilde{\mathsf{k}}^{l^*}_{i^*} = 0$. It outputs the resulting transcript as the view of MIM.
3. Next, the function f_{MIM} on input L, sets $L^{l^*}_{i^*,j^*} = L, R^{l^*}_{i^*,j^*} = 0$. The function has hardwired values $(\{\tilde{K}_l, \tilde{c}^l_1, \ldots, \tilde{c}^l_\lambda, \tilde{\mathsf{k}}^l_1, \ldots, \tilde{\mathsf{k}}^l_\lambda\}_{l\in[\ell]})\{(\mathsf{vk}^l, x^l_j, \mathsf{com}_{L^l,i,j}, \mathsf{com}_{R^l,i,j}, (\mathsf{dec}_{L^l,i,j}, s_{L^l,i,j}), (\mathsf{dec}_{R^l,i,j}, s_{R^l,i,j})\}_{i\in[\lambda],j\in[m],l\in[\ell]}$ as well as the values and $(L^l_{i,j}, R^l_{i,j}, s'_{L^l,i,j}, s'_{R^l,i,j})_{(j,l,i)\in[m]\times[\ell]\times[\lambda]\setminus\{i^*,j^*,k^*\}}$. The function sets the value $s'_{L^{l^*},i^*,j^*} = s_{L^{l^*},i^*,j^*} \oplus L$ and $s'_{R^{l^*},i^*,j^*}$ as a random string. The function using the hardwired values and $s'_{L^{l^*},i^*,j^*}, s'_{R^{l^*},i^*,j^*}$ computes the output out for the honest senders in the all ℓ left executions. It then invokes $\Pi.\mathsf{Sim}^2$ on out to generate the third round message of the protocol transcript in the main thread if $\tilde{c}^{l^*}_{i^*} \oplus \tilde{\mathsf{k}}^{l^*}_{i^*} = 0$, and generates the third round message of the protocol transcript in the rewinding thread if $\bar{c}^{l^*}_{i^*} \oplus \tilde{\mathsf{k}}^{l^*}_{i^*} = 0$. Using the opening information $\{(\tilde{\mathsf{dec}}_{L,\alpha_j,j}, \tilde{s}_{L,\alpha_j,j})\}_{j\in[m],l\in[\ell]}$ or $\{(\tilde{\mathsf{dec}}_{R,\alpha_j,j}, \tilde{s}_{R,\alpha_j,j})\}_{j\in[m],l\in[\ell]}$ and the values $\{\tilde{s'}_{L^l,\alpha_j,j}, \tilde{s'}_{R^l,\alpha_j,j}\}_{j\in[m],l\in[\ell]}$ obtained from the MIM, it computes and output the values $\{\tilde{L}^l_{\alpha_j,j}\}_{j\in[m],l\in[\ell]}$ or $\{\tilde{R}^l_{\alpha_j,j}\}_{j\in[m],l\in[\ell]}$.

4. Next, the function g_{MIM} on input L, sets on input R, sets $L^{l^*}_{i^*,j^*} = 0, R^{l^*}_{i^*,j^*} = R$. The function has hardwired values $(\{\tilde{K}_l, \tilde{c}^l_1, \dots, \tilde{c}^l_\lambda, \tilde{\mathsf{k}}^l_1, \dots, \tilde{\mathsf{k}}^l_\lambda\}_{l\in[\ell]}), \{(\mathsf{vk}^l, x^l_j, \mathsf{com}_{L^l,i,j}, \mathsf{com}_{R^l,i,j}, (\mathsf{dec}_{L^l,i,j}, s_{L^l,i,j}), (\mathsf{dec}_{R^l,i,j}, s_{R^l,i,j})\}_{i\in[\lambda],j\in[m],l\in[\ell]}$ as well as the values $(L^l_{i,j}, R^l_{i,j}, s'_{L^l,i,j}, s'_{R^l,i,j})_{(j,l,i)\in[m]\times[\ell]\times[\lambda]\setminus\{i^*,j^*,k^*\}}$. The function sets the value $s'_{R^{l^*},i^*,j^*} = s_{R^{l^*},i^*,j^*} \oplus R$ and $s'_{L^{l^*},i^*,j^*}$ as a random string. The function using the hardwired values and $s'_{L^{l^*},i^*,j^*}, s'_{R^{l^*},i^*,j^*}$ computes the output out for the honest senders in the all ℓ left executions. It then invokes $\Pi.\mathsf{Sim}^2$ on out to generate the third round message of the protocol transcript in the main thread if $\tilde{c}^{l^*}_{i^*} \oplus \tilde{\mathsf{k}}^{l^*}_{i^*} = 1$, and generates the third round message of the protocol transcript in the rewinding thread if $\vec{c}^{l^*}_{i^*} \oplus \tilde{\mathsf{k}}^{l^*}_{i^*} = 1$. Using the opening information $\{(\tilde{\mathsf{dec}}_{L,\alpha_j,j}, \tilde{s}_{L,\alpha_j,j})\}_{j\in[m],l\in[\ell]}$ or $\{(\tilde{\mathsf{dec}}_{R,\alpha_j,j}, \tilde{s}_{R,\alpha_j,j})\}_{j\in[m],l\in[\ell]}$ and the values $\{\tilde{s'}_{L^l,\alpha_j,j}, \tilde{s'}_{R^l,\alpha_j,j}\}_{j\in[m],l\in[\ell]}$ obtained from the MIM, it computes and output the values $\{\tilde{L}^l_{\alpha_j,j}\}_{j\in[m],l\in[\ell]}$ or $\{\tilde{R}^l_{\alpha_j,j}\}_{j\in[m],l\in[\ell]}$.

By Definition 2.7 of 1-many non-malleable codes,

$$(L, \mathsf{NM.Decode}(f_{\mathsf{MIM}}(L), g_{\mathsf{MIM}}(R))|(L, R \leftarrow \mathsf{NM.Code}(x^{l^*}_{j^*}))) \approx_\varepsilon$$
$$(L, \mathsf{NM.Decode}(f_{\mathsf{MIM}}(L), g_{\mathsf{MIM}}(R))|(L, R \leftarrow \mathsf{NM.Code}(0))) \approx_\varepsilon$$
and
$$(L, \mathsf{NM.Decode}(g_{\mathsf{MIM}}(L), f_{\mathsf{MIM}}(R))|(L, R \leftarrow \mathsf{NM.Code}(x^{l^*}_{j^*}))) \approx_\varepsilon$$
$$(L, \mathsf{NM.Decode}(g_{\mathsf{MIM}}(L), f_{\mathsf{MIM}}(R))|(L, R \leftarrow \mathsf{NM.Code}(0))) \approx_\varepsilon$$

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

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

Setting $h = h_{\mathsf{MIM}}$ for f_{MIM} and g_{MIM} defined above, these distributions correspond to the outputs of $\mathsf{H}^{l^*}_{4,i^*-1,j^*}$ (respectively $\mathsf{H}^{l^*}_{4,i^*,j^*}$) respectively, whenever $\tilde{c}^{l^*}_{i^*} \oplus \tilde{\mathsf{k}}^{l^*}_{i^*} \neq \tilde{c}^{l^*}_{i^*} \oplus \tilde{\mathsf{k}}^{l^*}_{i^*}$, the distributions $\mathsf{H}^{l^*}_{4,i^*-1,j^*}$ and $\mathsf{H}^{l^*}_{4,i^*,j^*}$ are $\varepsilon(\lambda)$-statistically indistinguishable because they jointly only depend on one of the shares, L or R. Since $\varepsilon(\lambda) = \mathrm{negl}(\lambda)$, this concludes the proof.

□

Receiver Privacy. We prove now that the protocol in Figure 8.5 satisfies receiver privacy against a corrupted sender S^*. The simulator Sim for the receiver privacy runs the simulator of $\Pi.\mathsf{Sim} = (\Pi.\mathsf{Sim}^1, \Pi.\mathsf{Sim}^2)$ against a corrupted sender. In more detail, Sim starts $\Pi.\mathsf{Sim}^1$ and acts as a proxy for the messages of Π between her and S^* (rewinding S^* when $\Pi.\mathsf{Sim}^1$ asks so). Moreover, Sim acts as an ideal functionality for $\Pi.\mathsf{Sim}^1$ and upon receiving the extracted sender inputs $\{x^*_i\}_{i\in[m]}$ from S^* she invokes the ideal functionality F^m_{OT} on $\{x^*_i\}_{i\in[m]}$. At this point Sim runs $\Pi.\mathsf{Sim}^2$ and acts as a proxy for the messages of Π between her and S^* (rewinding S^* when $\Pi.\mathsf{Sim}^2$ asks so). Sim gives in output the output of $\Pi.\mathsf{Sim}^2$.

The indistinguishability between the real game where S^* acts with an honest receiver R and the simulated game where S^* acts with Sim follows from the receiver security of Π. Moreover, since $\Pi.\mathsf{Sim}$ runs in polynomial time so does Sim.

□

8.3 List-Simultaneous Multiparty OT

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

Formally, we define the ideal (multiparty) simultaneous OT functionality $\mathcal{F}_{\mathsf{OT}}^{n\cdot(n-1)}$. this consists of $n(n-1)$ independent instances of α-out-of-β OTs, one for each ordered pair $(i,j) \in [n] \times [n]$ such that $i \neq j$. The (i,j)-th OT instance obtains input $x_{i,j} = (x_{i,j}^1, \ldots, x_{i,j}^\beta)$ form sender P_i and input $y_{i,j} \subseteq [\beta]$ with $|y_{i,j}| = \alpha$ from the receiver P_j. The functionality then outputs $\{x_{i,j}^k\}_{k \in y_{j,i}}$ to P_j and outputs $\perp$ to P_i for each $i,j \in \{[n] \times [n]\}$ with $i \neq j$.

Definition 8.7 (Simultaneous OT Protocol). *A (multiparty) simultaneous OT protocol* mpOT *is defined by a tuple of algorithms* $(\mathsf{mpOT}_1, \mathsf{mpOT}_2, \mathsf{mpOT}_3, \mathsf{out}_{\mathsf{mpOT}})$. *For each round $r \in [4]$, the i-th party in the protocol runs* mpOT_r *on 1^λ, the index i, the private input $\{x_{i,j}, y_{i,j}\}_{i \neq j}$ and the transcript of the protocol in the first $(r-1)$ rounds to obtain* mpOT_r^i. *It sends* mpOT_r^i *to every other party via a broadcast channel. We let* $\mathsf{mpOT}(r)$ *denote the transcript of the protocol* mpOT *in the first r rounds. The output computing function* $\mathsf{out}_{\mathsf{mpOT}}$ *takes the index i of a party, its private input, its random tape, and the transcript* $\mathsf{mpOT}(3)$ *and generates the output* out_i. *The protocol is required to satisfy the following property.*

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

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

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

In terms of security, we consider a slight variation of the notion of list MPC that we have recalled in Definition 3.2. In particular, in this notion we have two types of inputs: *static*, and *list*. The static inputs are inputs decided at the onset of the ideal/real-world experiment, whereas the list-inputs are the input sampled by the ideal functionality (on behalf of the honest parties) by using some samplers $\{\mathcal{X}_i\}_{i \in [H]}$. In the specific case of the OT functionality we consider here, the inputs of the honest receivers are fixed, whereas the inputs of the honest senders are sampled from the ideal functionality in the spirit of Definition 3.2. We provide the formal definition of the ideal and real-world in Figure 8.7, and we say that a simultaneous OT protocol is secure if it satisfies the following.

Definition 8.8 (Secure Simultaneous OT Protocol).

Security: *Let $\mathcal{A}$ be an adversary corrupting an arbitrary subset of parties indexed by M (we denote the indices of the honest parties with $H = [n] \backslash M$), that obtains auxiliary input* aux. *Let* $\{\mathsf{Real}_{\mathcal{A},\mathsf{mpOT}}(1^\lambda, M, H, \{y_{i,j}\}_{i \in H, j \in M})\}_\lambda$ *denote the joint distribution of the view of the adversary and the outputs of honest parties in the real-world world experiment described in Figure 8.7. We require the existence of an expected polynomial time simulator* Sim *that with black-box access to the adversary $\mathcal{A}$ interacts with the ideal functionality $\mathcal{F}^{\mathsf{list}}_{OT^{\alpha,\beta}}$ (see Figure 8.6) as described in the ideal world experiment of Figure 8.7, and return an output, denoted by* $\{\mathsf{Ideal}_{\mathsf{Sim},\mathcal{F}^{\mathsf{list}}_{OT^{\alpha,\beta}}}(1^\lambda, M, H, \{y_{i,j}\}_{i \in H, j \in M})\}$ *such that the following holds for any $\{y_{i,j}\}_{i \in H, j \in M}$*

$$\begin{aligned}&\{\mathsf{Real}_{\mathcal{A},\mathsf{mpOT}}(1^\lambda, M, H, \{y_{i,j}\}_{i \in H, j \in M})\}_{\lambda \in \mathbb{N}} \\ \approx_c &\{\mathsf{Ideal}_{\mathsf{Sim},\mathcal{F}^{\mathsf{list}}_{OT^{\alpha,\beta}}}(1^\lambda, M, H, \{y_{i,j}\}_{i \in H, j \in M})\}_{\lambda \in \mathbb{N}}.\end{aligned}$$

Figure 8.6: The list functionality $\mathcal{F}^{\mathsf{list}}_{OT^{\alpha,\beta}}$

The functionality is parametrized by the samplers $\{\mathcal{X}_i\}_{i\in H}$, and by the honest receivers inputs $\{y_{i,j}\}_{i\in H, j\in M}$, where $y_{i,j} \subseteq [\beta]$, with $|y_{i,j}| = \alpha$ for each $i \in H, j \in M$.

1. Upon receiving the receivers-inputs of the corrupted parties $\{y_{i,j}\}_{i\in M, j\in H}$, and $k \in \mathbb{N}$, for each $\kappa \in [k]$ do the following
   - For each $i \in H, j \in M$ sample $x_{i,j,\kappa} \leftarrow \mathcal{X}_i$, where $x_{i,j,\kappa} = (x^1_{i,j,\kappa}, \dots, x^\beta_{i,j,\kappa})$, and compute $\mathsf{out}^\kappa_{i,j} := \{x^c_{i,j,\kappa}\}_{c\in y_{j,i}}$
2. Send $\{\mathsf{out}^\kappa_{i,j}\}_{\kappa\in[k], i\in H, j\in M}$ to $\mathcal{A}$
3. Upon receiving $(\iota, \{x_{i,j}\}_{i\in M, j\in H}, \mathsf{abort})$ from $\mathcal{A}$ do the following. For each $j \in H \setminus \mathsf{abort}$ send $\{x^c_{i,j}\}_{i\in M, c\in y_{j,i}}$ and $\{x_{j,i,\iota}\}_{i\in M}$ to the honest P_j. For each $j \in \mathsf{abort}$ send an abort command to P_j.

Figure 8.7: Real and ideal world

$\mathsf{Real}_{\mathcal{A},\mathsf{mpOT}}(1^\lambda, M, H, \{y_{i,j}\}_{i\in H, j\in M})$

1. For each $i \in H$ sample $\{x_{i,j}\}_{j\in[M]} \leftarrow \mathcal{X}^i$.
2. For each $i \in H$ compute the first round π^i_1 of mpOT as the honest P_i would do on input $(\{x_{i,j}, y_{i,j}\}_{j\in[n]\setminus\{i\}})$ and the randomness $r_i \leftarrow \{0,1\}^\lambda$ and send π^i_1 to $\mathcal{A}$.
3. Upon receiving $\{\pi^i_1\}_{i\in M}$ from $\mathcal{A}$, for each $i \in H$ compute the second round π^i_2 of mpOT as the honest party P_i thus obtaining π^i_2 and send $(\pi^i_2, \{x_{i,j}\}_{j\in M})$ to $\mathcal{A}$.
4. Upon receiving $\{\pi^i_2\}_{i\in M}$ from $\mathcal{A}$, for each $i \in H$ compute π^i_3 as the honest party P_i would do thus obtaining π^i_3, and send π^i_3 to $\mathcal{A}$.
5. Upon receiving $\{\pi^i_3\}_{i\in M}$, compute the output out_i as P_i would do, and return the view of $\mathcal{A}$.

$\mathsf{Ideal}_{\mathsf{Sim},\mathcal{F}^{\mathsf{list}}_{OT^{\alpha,\beta}}}(1^\lambda, M, H, \{y_{i,j}\}_{i\in H, j\in M})$

1. $(\{y_{i,j}\}_{i\in M, j\in H}, 1^k, z) \leftarrow \mathsf{Sim}^{\mathcal{A}}_1(1^\lambda)$
2. Send $\{y_{i,j}\}_{i\in M, j\in H}$ to the ideal functionality $\mathcal{F}^{\mathsf{list}}_{OT^{\alpha,\beta}}$ which computes $\{\mathsf{out}^\kappa_{i,j}\}_{i\in H, j\in M, \kappa\in[k]}$ as described using the sampled inputs $\{x_{i,j,\kappa}\}_{i\in[H], j\in M, \kappa\in[k]}$ as described before.
3. Whenever $\mathsf{Sim}^{\mathcal{A}}_2(\{\mathsf{out}^\kappa_{i,j}\}_{i\in M, j\in H, \kappa\in[k]}, z)$ queries $\mathcal{A}$ with a message $(\{\pi_i\}_{i\in H}, \gamma)$ (with $\gamma \in [k]$), replace the query with $(\{\pi_i\}_{i\in H}, \{x_{i,j,\gamma}\}_{i\in[H], j\in M}, \gamma)$ and forward the pair to $\mathcal{A}$.
4. $(\iota, \{x_{i,j}\}_{i\in[M], j\in H}, \mathrm{View}) \leftarrow \mathsf{Sim}^{\mathcal{A}}_2(\{\mathsf{out}^\kappa_{i,j}\}_{i\in M, j\in H, \kappa\in[k]}, z)$ with $\iota \in [k] \cup \{\perp\}$.
5. Send $(\iota, \{x_{i,j}\}_{i\in[M], j\in H}, \mathsf{abort})$ where abort is a set of indices that determines which honest parties should not receive the output from $\mathcal{F}^{\mathsf{list}}_{OT^{\alpha,\beta}}$.
6. Return View.

Remark 8.9 (Watchlist Protocol). The watchlist protocol is defined identically to the mpOT protocol (Definition 8.7).

Following the approach proposed in [IKSS21], to realize the watchlist protocol we let each pair of parties P_i, P_j interact in an execution of non-malleable OT where in this execution P_i is acting as the sender and P_j as receiver, for each $j, i \in [n]$ with $i \neq j$. This allows each party to get an OT correlation with the other party acting as a sender and each other party to obtain an OT correlation with each other party behaving as the receiver.

More formally, the multiparty simultaneous OT mpOT works as follows:

– For $i \in [n]$, let P_i's OT input be $(\{X_{i,j}\}_{j\in[n]\setminus\{i\}}, \{y_{i,j}\}_{j\in[n]\setminus\{i\}})$, where:
  - For every $j \in [n] \setminus \{i\}, y_{i,j} \subset [m]$ such that $|y_{i,j}| = k$, and
  - For every $j \in [n] \setminus \{i\}, X_{i,j} = (s_{i,j,1}, \dots, s_{i,j,m})$.
– Furthermore, we denote an instance of the ℓ non-malleable m-choose-k OT protocol by NM-OT.

Then, for every $i, j \in [n]$ with $i \neq j$, P_i and P_j execute an instance of NM-OT with P_i acting as the sender using $\{s_{i,j,k}\}_{k\in[m]}$ as its input and P_j acting as the receiver using $y_{j,i}$ as its input. The proof of security follows similar to the one of [IKSS21] and we provide a sketch of it below. More precisely we have the following theorem.

Theorem 8.10. *Let $H \subset [n]$ denote the subset of honest parties. Let $\{\mathcal{X}_i\}_{i\in H}$ be a high min-entropy random variable defined by a probability distribution $\{\mathcal{D}_i\}_{i\in H}$. Assuming that* NM-OT *satisfies Definition 8.1 (i.e.,* NM-OT *enjoys receiver security w.r.t. ideal functionality $\mathcal{F}^m_{\mathsf{OT}}$ and list non-malleability for functionality*

$\mathcal{F}^{\mathsf{list}}_{\mathsf{NMOT}}$ *parametrized by* $\{\mathcal{X}_i\}_{i\in H}$*) against sometimes aborting adversaries, then the* mpOT *described above satisfies Definition 8.7 for the functionality* $\mathcal{F}^{\mathsf{list}}_{OT^{\alpha,\beta}}$ *parametrized by* $\{\mathcal{X}_i\}_{i\in H}$*, against sometimes aborting adversaries. Moreover* mpOT *makes black-box use of* NM-OT*.*

Proof. Correctness of the n-party m-out-of-k (multiparty) simultaneous OT follows from the correctness of the m-out-of-k OT protocol.

We describe how to prove a stronger security property of the resulting (multiparty) simultaneous OT protocol: that for every corrupted subset $M \subset [n]$ and set of honest parties $H = [n] \setminus M$, there exists a simulator $\mathsf{Sim}_{\mathsf{mpOT}}$ such that for every (malicious) non-uniform adversary $\mathcal{A}$ that corrupts $\{P_i\}_{i\in M}$, and for every choice of inputs $\{y_{i,j}\}_{i\in H, j\in M}$ the following two distributions are computational indistinguishable:

$$\{\mathsf{Real}_{\mathcal{A},\mathsf{mpOT}}(1^\lambda, M, H, \{y_{i,j}\}_{i\in H,j\in M})\}_{\lambda\in\mathbb{N}}$$
$$\approx_c \{\mathsf{Ideal}_{\mathsf{Sim},\mathcal{F}^{\mathsf{list}}_{OT^{\alpha,\beta}}}(1^\lambda, M, H, \{y_{i,j}\}_{i\in H,j\in M})\}_{\lambda\in\mathbb{N}}.$$

Here, $\mathsf{Real}_{\mathcal{A},\mathsf{mpOT}}(1^\lambda, M, H, \{y_{i,j}\}_{i\in H,j\in M})$ denotes the adversary's view and the output of the honest parties in the real-world experiment defined in the Definition 8.7. Similarly $\{\mathsf{Ideal}_{\mathsf{Sim},\mathcal{F}^{\mathsf{list}}_{OT^{\alpha,\beta}}}(1^\lambda, M, H, \{y_{i,j}\}_{i\in H,j\in M})\}_{\lambda\in\mathbb{N}}$ denote adversary's view and the outputs of honest parties in the ideal execution defined in the Definition 8.7.

The simulator $\mathsf{Sim}_{\mathsf{mpOT}}$ is constructed as follows. It runs the simulator of the non-malleable OT NM-OT, $\mathsf{NM\text{-}OT.Sim} = (\mathsf{NM\text{-}OT.Sim}^1, \mathsf{NM\text{-}OT.Sim}^2)$ where $\mathcal{A}$ plays the role of MIM. In more details $\mathsf{Sim}_{\mathsf{mpOT}}$ invokes $\mathsf{NM\text{-}OT.Sim}^1$ on dummy inputs $[k]^{|H|\cdot|M|}$ acting as a proxy between $\mathcal{A}$ and $\mathsf{NM\text{-}OT.Sim}^1$ for the messages of mpOT. Upon receiving $(\{\tilde{y}_{i,j}\}_{j\in H,i\in M}, 1^\kappa)$ from $\mathsf{NM\text{-}OT.Sim}^1$, the simulator $\mathsf{Sim}_{\mathsf{mpOT}}$ forwards $(\{\tilde{y}_{i,j}\}_{j\in H,i\in M}, 1^\kappa)$ to the ideal functionality $\mathcal{F}^{\mathsf{list}}_{OT^{\alpha,\beta}}$ obtaining $\{\mathsf{out}^\kappa_{i,j}\}_{i\in M,\kappa\in[k]}$.

At this point $\mathsf{Sim}_{\mathsf{mpOT}}$ runs $\mathsf{NM\text{-}OT.Sim}^2$ on input $\{\mathsf{out}^\kappa_{i,j}\}_{i\in M,\kappa\in[k]}$ and acts with her as the ideal functionality $\mathcal{F}^{\mathsf{list}}_{\mathsf{NMOT}}$ would do. Moreover,$\mathsf{Sim}_{\mathsf{mpOT}}$ acts as a proxy between $\mathcal{A}$ and $\mathsf{NM\text{-}OT.Sim}^2$ for the messages of mpOT.

Upon receiving $(\iota, \text{View}, \{\tilde{x}_{i,j}\}_{i\in[M],j\in H})$ from $\mathsf{NM\text{-}OT.Sim}^2$, the simulator $\mathsf{Sim}_{\mathsf{mpOT}}$ sends $(\iota, \{\tilde{x}_{i,j}\}_{i\in[M],j\in H})$ to $\mathcal{F}^{\mathsf{list}}_{OT^{\alpha,\beta}}$ and output View.

We now define a sequence of hybrid experiments:

Hybrid H_0: This experiment corresponds to the real world execution of mpOT where the adversary $\mathcal{A}$ is interacting with $\{P_i\}_{i\in[n]\setminus M}$ which uses inputs $\{x_i = \{y_{i,j}, s_{i,j}\}_{j\in[n]\setminus\{i\}}\}_{i\in[n]\setminus M}$. The output of this experiment is the joint distribution of the view of the adversary $\mathcal{A}$ and the output of honest parties in this interaction.

Hybrid H_1: This experiment is defined as the previous one except that in this experiment the honest parties instead of using inputs $\{x_i = \{y_{i,j}, s_{i,j}\}_{j\in[n]\setminus\{i\}}\}_{i\in[n]\setminus M}$, they are using input $\{x'_i = \{[k], s_{i,j}\}_{j\in[n]\setminus\{i\}}\}_{i\in[n]\setminus M}$. Specifically, the input values are set to a dummy input $[k]$ instead of their real inputs $\{\{y_{i,j}\}_{j\in[n]\setminus\{i\}}\}_{i\in[n]\setminus M}$. The output of honest parties is generated identically to H_0. The output of this experiment is the joint distribution of the view of the adversary $\mathcal{A}$ and the output of honest parties in this interaction. The indistinguishability from H_0 follows from the receiver security of NM-OT.

Hybrid H_2: This experiment is defined as the previous one except that the simulator of the non-malleable OT is used. Specifically, the hybrid follows the same steps of $\mathsf{Sim}_{\mathsf{mpOT}}$. The output of this experiment is the joint distribution of the view of the adversary $\mathcal{A}$ and the output of honest parties in this interaction. The indistinguishability from H_1 follows from the list non-malleability of m-out-of-k NM-OT. Suppose that this is not the case, then we can show a reduction where the adversary $\mathcal{A}$ is playing as MIM that participates in at most $\ell(\lambda) = n(\lambda)^2$ sessions where an honest party acts as OT sender, and at most $\ell(\lambda) = n(\lambda)^2$ sessions where an honest party acts as OT receiver. Therefore, if $\mathcal{A}$ has a non-negligible advantage in distinguishing H_1 and H_2 then MIM has a non-negligible advantage in breaking the list non-malleability of NM-OT.

The hybrid H_0 corresponds the real-world and the hybrid H_2 to the ideal world. This concludes the proof. □

9 Four-Round Multiparty Computation

In this section, we present our MPC protocol. This is essentially an instantiation of the IPS compiler [IPS08], following the approach of [IKSS21]. In a nutshell, the IPS compiler works as follows: a client-server protocol Φ, with a security threshold of $1/3$ is being executed to evaluate the function f that should be computed by the MPC protocol, this protocol is termed the outer protocol. To emulate the servers in this setting, additionally another semi-malicious MPC protocol, the inner-protocol $\mathit{\Pi}_h$, is executed in parallel to take over the role of the servers in the client-server protocol Φ. Due to the fact that the executed inner protocol is not maliciously secure, the overall protocol also does not achieve malicious security. To achieve full malicious security, we apply a cut-and-choose protocol. In more detail, we execute multiple instances of the inner protocol $\mathit{\Pi}_h$ and run a cut-and-choose protocol over these instances, which prevents and adversary from the malicious generation of messages. The cut-and-choose is realized using a watchlist protocol as described in the previous section.

More formally, our construction makes use of the following building blocks.

– A MAC scheme MAC.
– The Outer Protocol $\Phi = (\Phi_1, \Phi_2)$, a 2-round, n-client, m-server MPC protocol achieving privacy with knowledge of outputs (Remark 2.12) against a malicious, adaptive adversary corrupting up to $n-1$ clients and $t = (m-1)/3$ servers. We also need the protocol to be equipped with a simulator that works in two phases. In the first phase, it computes a first round of the honest parties accordingly to Φ_1 using default inputs. In the second phase Sim_Φ takes as input all the shares sent from the malicious client to the honest servers, all the shares sent from on the behalf of the honest clients to the corrupted servers, and generates its output by querying the ideal functionality using the inputs of the corrupted parties. Φ realizes the functionality g, wich takes the inputs $(z_1, \ldots, z_n)$, and does the following
  1. For each $i \in [n]$ parse z_i as x_1, k_i.
  2. Compute $y := f(x_1, \ldots, x_n)$.
  3. Compute $\sigma_i \coloneqq \mathsf{MAC}(\mathsf{k}_i, y)$ for all $i \in [n]$
  4. Return $(y, \sigma_1, \ldots, \sigma_n)$.

  In our protocol, we set $t = 2\lambda n^2$ and refer to Section 2.9 for more detail about the outer protocol we use. All the properties required by our outer protocol are satisfied by the construction proposed in [IKP10].
– The inner protocol $\Pi_h = (\Pi_{h,1}, \Pi_{h,2}, \Pi_{h,3})$ realized in Appendix A. Let Us and Cs be two disjoint subsets of $[m]$ with $|\mathsf{Cs}| \leq t$ and $|\mathsf{Us}| - |\mathsf{Cs}| = m$. In our construction, we will consider parallel executions of the inner protocol, and an ideal world parametrized by $\mathcal{S}_{\mathsf{out}}$. $\mathcal{S}_{\mathsf{out}}$ takes as input $\{\phi_1^{i\to h}\}_{i\in M, h\in \mathsf{Us}}$ $\{\phi_1^{i\to h}\}_{i\in H, h\in \mathsf{Cs}}$, where, for $h \in \mathsf{Us}$, $\{\phi_1^{i\to h}\}_{i\in M}$ represents the inputs used by corrupted parties in the h-th execution of the inner protocol, and for $h \in \mathsf{Cs}$, $\{\phi_1^{i\to h}\}_{i\in H, h\in\mathsf{Cs}}$ represents the inputs used by the honest parties in the h-th execution of the inner protocol. On these inputs, $\mathcal{S}_{\mathsf{out}}$ runs the second phase of Sim_Φ thus obtaining ϕ_2^h, for each $h \in \mathsf{Us}$.
– A list-simulatable secure Watchlist Protocol $\mathsf{WL} = (\mathsf{WL}_1, \mathsf{WL}_2, \mathsf{WL}_3)$ as realized in Section 8 for the functionality $\mathcal{F}_{\mathsf{OT}}^{\mathsf{list},\mathcal{X}}$ where $\mathcal{X}$ is defined by all the possible outputs that can be returned by the following process ($\mathcal{X}$ is parametrized by the input of the honest parties x_i for each $i \in H$, where H denotes the indices-set of the honest parties).
  1. Initialize the empty list Y. For each $i \in H$
     (a) Choose a random MAC key $\mathsf{k}_i \leftarrow \{0,1\}^*$ and set $z_i \coloneqq (x_i, \mathsf{k}_i)$.
     (b) Compute $(\phi_1^{i\to 1}, \ldots, \phi_1^{i\to m}) \leftarrow \Phi_1(1^\lambda, i, z_i)$.
     (c) Sample $r_{i,h} \leftarrow \{0,1\}^*$ for all $h \in [m]$ and set $y_{i,j} \coloneqq \{r_{i,h}, \phi_1^{i\to h}\}$.
     (d) add $y_{i,j}$ to Y.
  2. return $Z = \{Y, \ldots, Y\}$ with $|Z| = n$.

  Here, we denote by $\mathsf{wl}(r)$ the transcript in the first r rounds of WL.
  We refer to Figure 9.1 for the formal description of the protocol.

Figure 9.1: MPC

Initialization: Each party P_i uses x_i as its input.

Round 1.

1. Choose a random MAC key $\mathsf{k}_i \leftarrow \{0,1\}^*$ and sets $z_i := (x_i, \mathsf{k}_i)$.
2. Compute $(\phi_1^{i\to 1}, \ldots, \phi_1^{i\to m}) \leftarrow \Phi_1(1^\lambda, i, z_i)$.
3. Sample a random subset $K_I \subset [m]$ of size λ and set $x_{i,j} := K_i$ for all $j \in [n] \setminus \{i\}$.
4. Sample $r_{i,h} \leftarrow \{0,1\}^*$ for all $h \in [m]$ and set $y_{i,j} := \{r_{i,h}, \phi_1^{i\to h}\}$ for all $j \in [n] \setminus \{i\}$.
5. Compute $\mathsf{wl}_1^i \leftarrow \mathsf{WL}_1(1^\lambda, i, \{x_{i,j}, y_{i,j}\}_{j\in[n]\setminus\{i\}}, \mathsf{wl}(0))$.
6. Broadcast wl_1^i.

Round 2.

1. Compute $\mathsf{wl}_2^i \leftarrow \mathsf{WL}_2(1^\lambda, i, \{x_{i,j}, y_{i,j}\}_{j\in[n]\setminus\{i\}}, \mathsf{wl}(1))$.
2. Compute $\pi_{h,1}^i := \Pi_{h,1}(1^\lambda, i, \phi_1^{i\to h}; r_{i,h})$ for all $h \in [m]$.
3. Broadcast $\mathsf{wl}_2^i, \{\pi_{h,1}^i\}_{h\in[m]}$.

Round 3.

1. Compute $\mathsf{wl}_3^i \leftarrow \mathsf{WL}_3(1^\lambda, i, \{x_{i,j}, y_{i,j}\}_{j\in[n]\setminus\{i\}}, \mathsf{wl}(2))$.
2. Compute $\pi_{h,2}^i := \Pi_{h,2}(1^\lambda, i, \phi_1^{i\to h}, \pi_h(1); r_{i,h})$ for all $h \in [m]$.
3. Broadcast $\mathsf{wl}_3^i, \{\pi_{h,2}^i\}_{h\in[m]}$.

Round 4.

1. Run $\mathsf{out}_{\mathsf{WL}}$ on $(i, \{x_{i,j}, y_{i,j}\}_{j\in[n]\setminus\{i\}})$, the random tape and $\mathsf{wl}(4)$ to obtain $\{r_{j,h}, \phi_1^{j\to h}\}_{j\in[n]\setminus\{i\}, h\in K_i}$.
2. For all $j \in [n] \setminus \{i\}$ and $h \in K_i$, check that:
   - The PRG computations in $\phi_1^{j\to h}$ are correct.
   - For all $\ell \in [2]$, whether $\pi_{h,\ell}^j = \Pi_{h,\ell}(1^\lambda, j, \phi_1^{j\to h}, \pi_h(\ell-1); r_{j,h})$.
3. If any of the above checks fail, output $\perp$.
4. For all $h \in [m]$, compute $\pi_{h,3}^i := \Pi_{h,3}(1^\lambda, i, \phi_1^{i\to h}, \pi_h(2); r_{i,h})$.
5. Broadcast $\{\pi_{h,3}^i\}_{h\in[m]}$.

Output Computation.

1. Compute $\phi_2^h := \mathsf{out}_{\Pi_h}(i, \pi(3))$ for all $h \in [m]$.
2. Compute $(y, \sigma_1, \ldots, \sigma_n) := \mathsf{out}_\Phi(\{\phi_2^h\}_{h\in[m]})$.
3. Check if σ_i is a valid tag on y using the key k_i. If the check is successful output y, otherwise output $\perp$.

Theorem 9.1. *Let* wl *be the watchlist protocol for the high min-entropy random variable* $\mathcal{X}$ *defined above that samples randomly from* Φ_1 *and* $\{0,1\}^\lambda$*, let* Π_h *be a secure inner-protocol,* Φ *a secure outer protocol and* MAC *a secure MAC scheme, then the protocol* Π *is secure against sometimes aborting adversaries.*

To prove the security of this protocol, we first present the corresponding simulator and afterwards describe the different hybrids and argue their indistinguishability.

Simulator. Let $\mathcal{A}$ be an adversary that corrupts the set of parties indexed by M and let $H := [n] \setminus M$. The simulator of Π_{mpc} then works as follows:

Figure 9.2: Simulator Sim of Π_{mpc}

Let $(\mathsf{Sim}_{\mathsf{wl}}^1, \mathsf{Sim}_{\mathsf{wl}}^2)$ denote the simulator for the watchlist protocol. Let Sim_{Π_h} denote the simulator for the h-th inner-protocol, and let Sim_Φ denote the simulator for the outer protocol $\Phi = (\Phi_1, \Phi_2, \mathsf{out}_\Phi)$.

1) First message & extraction of the input: Sim behaves as follows:

1. Run the simulator $\mathsf{Sim}_{\mathsf{wl}}^1$ of the watchlist protocol. This simulator requires the generation of multiple second and third rounds to estimate the abort probability of the adversary. This, in turn, requires the generation of messages w.r.t. the protocol Π_h, which in this step of the simulation are honestly generated using shares, returned by executions of Φ on random inputs.
2. Afterwards, the simulator $\mathsf{Sim}_{\mathsf{wl}}^1$ will output the extracted inputs of the adversary $\mathcal{A}$ where it acts as a receiver $\{x_{i,j}\}_{i\in M,j\in H}$, the inputs of the adversary $\mathcal{A}$ where it acts as a sender $\{y_{i,j}\}_{i\in M,j\in H}$ and the integer d. Note that $\{y_{i,j}\}_{i\in M,j\in H}$ represents the input-randomness pairs the adversary has used to compute the first second round of Π_h.
3. Set $C \coloneqq \{x_{i,j}\}_{i\in M,j\in H}$.

2) Answering the ideal watchlist functionality query: Sim answers the ideal functionality query that is asked by $\mathsf{Sim}_{\mathsf{wl}}$ in the following way:
1. Run Φ on random inputs d-times to obtain $\{\phi_1^{i\to j,k}\}_{i\in H,j\in[m]}$ for all $k\in[d]$.
2. When $\mathsf{Sim}_{\mathsf{wl}}$ submits its ideal functionality query $(\{x_{i,j}\}_{i\in M,j\in H}, 1^d)$ answer with the tuple $(\{\phi^{i\to h,1}, r_h^{i,1}\}_{h=x_{i,j},x_{i,j}\in C}, \ldots, \{\phi^{i\to h,d}, r_h^{i,d}\}_{h=x_{i,j},x_{i,j}\in C})$.

3) Forcing the output: Sim behaves as follows for the output of the watchlist protocol:
1. When $\mathsf{Sim}_{\mathsf{wl}}^2$ is forcing the output $\{\phi^{i\to h,k}, r_h^{i,k}\}_{h=x_{i,j},x_{i,j}\in C})$ with $k\in[d]$, for each $h\in[m]$, $i\in H$ act as follows with respect to the inner-protocol messages

   Second round (a) If $h\in C$ then instruct Sim_{Π_h} to corrupt the i-th party and give $(\phi_1^{i\to h,k}, r_{h,1}^{i,k})$ as its corresponding input-randomness pair. Let $\pi_{h,1}^{i,k}$ be the output computed by this party.
   (b) If $h\notin C$ define the oracle $\mathcal{D}^{h,k}$ which on input i returns $\phi_1^{i\to h,k}$. For each $i\in H$ if $h\notin C$ compute $(\pi_{h,1}^{i,k}, \mathsf{st}_{h,1}^{i,k}) \coloneqq \mathsf{Sim}_{\Pi_h}(1^\lambda, i, 0^\lambda)$,
   (c) Send $\pi_{h,1}^{i,k}$ to complete the second round.

   Third round (a) If $h\in C$ then compute the second round of Π_h with respect to the i-th party using the input-randomness pair $(\phi_1^{i\to h,k}, r_{h,1}^{i,k})$. Let $\pi_{h,2}^{i,k}$ be the output computed by this party.
   (b) If $h\notin C$ compute $(\pi_{h,2}^{i,k}, \phi_1^{i\to h,k}) \coloneqq \mathsf{Sim}_{\Pi_h}^{\mathcal{D}^{h,k}}(1^\lambda, i, \pi_h(1), \mathsf{st}_{h,1}^{i,k})$.
   (c) Send $\pi_{h,2}^{i,k}$ to complete the third round.
2. After the simulator $\mathsf{Sim}_{\mathsf{wl}}^2$ has forced the output on the adversary $\mathcal{A}$, it will output the corresponding index $k\in[d]$ for which the forcing was successful. The following steps all proceed w.r.t. the k''th instance.

Sim checks the correctness of the openings:
1. Initialize the empty set C'.
2. For each $h\in[m]$, check if there exists some $j\in H$ such that for every $i\in M, y_{i,j}$ contains the input and the randomness that explains the messages sent by corrupted parties in Π_h as well as the correct PRG computations. If not, it adds h to C'.
3. If such a j exists, then for every $i\in M$, use $(\phi_1^{i\to h,k}, r_{h,1}^{i,k})$ present in $y_{i,j}$ as the consistent input and randomness used by the corrupted party P_i in the protocol Π_h.

5) Further corruption of Φ based on opening: Sim distinguishes between two cases:
1. If $|C'| > \lambda n^2$, then Sim instructs the ideal functionality to send abort to all the honest parties and outputs the view of the adversary.
2. If $|C'| \le \lambda n^2$, then for each $i\in H$, Sim chooses a random subset K_i of $[m]$ of size λ. If for any $h\in K_i, \{y_{j,i}\}_{j\in M}$ contains inconsistent input and randomness, then Sim instructs the ideal functionality to send abort to i. We denote by H' the subset of honest parties $H'\subset H$ of honest parties that did not abort.

6) Gerneration of the fourth round:
1. When $\mathcal{S}_{\mathsf{out}}$ sends the query $\{z_i \coloneqq (x_i, \mathsf{k}_i)\}_{i\in M}$ sends $\{x_i\}_{i\in M}$ to the ideal functionality f.
2. Upon receiving y, compute $\sigma_i \coloneqq \mathsf{MAC}(\mathsf{k}_i, y)$ for all $i\in[n]$ and send $(y, \sigma_1, \ldots, \sigma_n)$ to $\mathcal{S}_{\mathsf{out}}$.
3. Return whatever Sim_{Π_h} returns for every $h\in[m]$.

8) Output computation: Do as follows to compute the final output:
1. If $H' \neq H$, instruct the ideal functionality to output abort to all the honest parties.
2. For all $h \in [m]$, compute ϕ_2^h as $(y', \sigma'_1, \ldots, \sigma'_n) \coloneqq \mathsf{out}_{\Pi_h}(\pi_h(3))$.
3. Check if $y' = y$ and for each $i \in H$ that $\sigma'_i = \sigma_i$. For every $i \in H$, such that the above checks pass, the ideal functionality is instructed to deliver the outputs to P_i. All other parties are instructed to abort.

Sim also keeps a count of its overall running time and if it reaches 2^λ steps it outputs fail.

Running time of the Simulator. The running time of this simulator depends on the running time of the simulator for the watchlist protocol $\mathsf{Sim}_{\mathsf{wl}}$, the simulator for the inner-protocol Sim_{Π_h} and the simulator for the outer protocol Φ. Since all of these simulators run in expected polynomial time, also the simulator Sim runs in expected polynomial time.

Hybrid Transactions Now, we prove the security of the previously described protocol and start by introducing the different hybrids:

Hybrid H_0: This hybrid corresponds to the execution of the protocol in the real world.

Hybrid H_1: In this hybrid, the watchlist protocol wl is not executed honestly anymore but simulated, i.e. the simulator $\mathsf{Sim}_{\mathsf{wl}} = (\mathsf{Sim}^1_{\mathsf{wl}}, \mathsf{Sim}^2_{\mathsf{wl}})$ is run. $\mathsf{Sim}_{\mathsf{wl}}$ interacts with an ideal functionality parametrized by $\mathcal{X}$. In the hybrid, the messages of the inner protocol executions are computed accordingly to what the simulator $\mathsf{Sim}_{\mathsf{wl}}$ samples from $\mathcal{X}$. The indistinguishability between this and the previous hybrid follows from the security of the watchlist protocol wl and is proven in Lemma 9.2.

Hybrid H_2: In this hybrid, the set C' is defined as described in the simulator and if the size of C' is bigger than λn^2, the honest parties abort. The indistinguishability between this and the previous hybrid is proven in Lemma 9.3.

Hybrid H_3: In this hybrid, the messages of the inner-protocol Π_h executions are simulated. The indistinguishability between this and the previous hybrid follows from the security of the inner-protocol Π_h and the security of the outer protocol Φ. We refer to Figure 9.3 for the formal description of the hybrid, and to Lemma 9.4 for the formal proof.

Figure 9.3: Hybrid H_3

1) First message & extraction of the input: Sim behaves as follows:
1. Run the simulator $\mathsf{Sim}^1_{\mathsf{wl}}$ (recall that the ideal world for the watchlist protocol is parametrized by $\mathcal{X}$ as defined in H_1) until it returns the extracted inputs of the adversary $\mathcal{A}$ where it acts as a receiver $\{x_{i,j}\}_{i \in M, j \in H}$, the inputs of the adversary $\mathcal{A}$ where it acts as a sender $\{y_{i,j}\}_{i \in M, j \in H}$ and the integer d. Note that $\{y_{i,j}\}_{i \in M, j \in H}$ represents the input-randomness pairs the adversary has used to compute the first second round of Π_h.
2. Set $C \coloneqq \{x_{i,j}\}_{i \in M, j \in H}$.

2) Answering the ideal watchlist functionality query: Sim answers the ideal functionality query that is asked by $\mathsf{Sim}_{\mathsf{wl}}$ in the following way:
1. Run Φ using the honest parties' inputs d-times (using fresh randomness every time) thus obtaining $\{\phi_1^{i \to j,k}\}_{i \in H, j \in [m]}$ for all $k \in [d]$.
2. When $\mathsf{Sim}_{\mathsf{wl}}$ submits its ideal functionality query $(\{x_{i,j}\}_{i \in M, j \in H}, 1^d)$ answer with the tuple $(\{\phi^{i \to h,1}, r_h^{i,1}\}_{h = x_{i,j}, x_{i,j} \in C}, \ldots, \{\phi^{i \to h,d}, r_h^{i,d}\}_{h = x_{i,j}, x_{i,j} \in C})$.

3) Forcing the output: Sim behaves as follows for the output of the watchlist protocol:
1. When $\mathsf{Sim}^2_{\mathsf{wl}}$ is forcing the output $\{\phi^{i \to h,k}, r_h^{i,k}\}_{h = x_{i,j}, x_{i,j} \in C})$ with $k \in [d]$, for each $h \in [m]$, $i \in H$ act as follows with respect to the inner-protocol messages

   Second round (a) If $h \in C$ then instruct Sim_{Π_h} to corrupt the i-th party and give $(\phi_1^{i \to h,k}, r_{h,1}^{i,k})$ as its corresponding input-randomness pair. Let $\pi_{h,1}^{i,k}$ be the output computed by this party.

(b) If $h \notin C$ define the oracle $\mathcal{D}^{h,k}$ which on input i returns $\phi_1^{i \to h,k}$. For each $i \in H$ if $h \notin C$ compute $(\pi_{h,1}^{i,k}, \mathsf{st}_{h,1}^{i,k}) \coloneqq \mathsf{Sim}_{\Pi_h}(1^\lambda, i, 0^\lambda)$,

(c) Send $\pi_{h,1}^{i,k}$ to complete the second round.

Third round (a) If $h \in C$ then compute the second round of Π_h with respect to the i-th party using the input-randomness pair $(\phi_1^{i \to h,k}, r_{h,1}^{i,k})$. Let $\pi_{h,2}^{i,k}$ be the output computed by this party.

(b) If $h \notin C$ compute $(\pi_{h,2}^{i,k}, \phi_1^{i \to h,k}) \coloneqq \mathsf{Sim}_{\Pi_h}^{\mathcal{D}^{h,k}}(1^\lambda, i, \pi_h(1), \mathsf{st}_{h,1}^{i,k})$.

(c) Send $\pi_{h,2}^{i,k}$ to complete the third round.

2. After the simulator $\mathsf{Sim}_{\mathsf{wl}}^2$ has forced the output on the adversary $\mathcal{A}$, it will output the corresponding index $k \in [d]$ for which the forcing was successful. The following steps all proceed w.r.t. the k''th instance.

Sim checks the correctness of the openings:
1. Initialize the empty set C'.
2. For each $h \in [m]$, check if there exists some $j \in H$ such that for every $i \in M, y_{i,j}$ contains the input and the randomness that explains the messages sent by corrupted parties in Π_h as well as the correct PRG computations. If not, it adds h to C'.
3. If such a j exists, then for every $i \in M$, use $(\phi_1^{i \to h,k}, r_{h,1}^{i,k})$ present in $y_{i,j}$ as the consistent input and randomness used by the corrupted party P_i in the protocol Π_h.

5) Further corruption of Φ based on opening: Sim distinguishes between two cases:
1. If $|C'| > \lambda n^2$, then make all the honest parties to abort.
2. If $|C'| \leq \lambda n^2$, then for each $i \in H$, Sim chooses a random subset K_i of $[m]$ of size λ. If for any $h \in K_i, \{y_{j,i}\}_{j \in M}$ contains inconsistent input and randomness, then Sim instructs the ideal functionality to send abort to i. We denote by H' the subset of honest parties $H' \subset H$ of honest parties that did not abort.

6) Gerneration of the fourth round:
1. When $\mathcal{S}_{\mathsf{out}}$ sends the query $\{z_i \coloneqq (x_i, \mathsf{k}_i)\}_{i \in M}$ compute $y \leftarrow f(x_1, \ldots, x_n)$ (i.e., use the honest inputs, and the inputs returned from $\mathcal{S}_{\mathsf{out}}$ to compute the output of the computation).
2. Compute $\sigma_i \coloneqq \mathsf{MAC}(\mathsf{k}_i, y)$ for all $i \in [n]$ and send $(y, \sigma_1, \ldots, \sigma_n)$ to $\mathcal{S}_{\mathsf{out}}$.
3. Return whatever Sim_{Π_h} returns for every $h \in [m]$.

8) Output computation: Do as follows to compute the final output:
1. If $H' \neq H$, instruct the ideal functionality to output abort to all the honest parties.
2. For all $h \in [m]$, compute ϕ_2^h as $(y', \sigma_1', \ldots, \sigma_n') \coloneqq \mathsf{out}_{\Pi_h}(\pi_h(3))$.
3. Return y'.

Sim also keeps a count of its overall running time and if it reaches 2^λ steps it outputs fail.

Hybrid H_4: In this hybrid the messages of the honest parties for the outer protocol Φ are computed with respect to a default input. I.e., in the rewinding threads and in the main thread, for each $i \in H$, the i-th party computes $(\phi_1^{i \to 1}, \ldots, \phi_1^{i \to m}) \leftarrow \Phi_1(1^\lambda, i, z_i)$ where $z_i = 0$. The indistinguishability between this and the previous hybrid comes from the security of the outer protocol and from the fact that we deal with a sometimes aborting adversary. We refer to Lemma 9.5

Hybrid H_5: In this hybrid we make the following change. In the output phase, we recover $(y', \sigma_1, \ldots, \sigma_1)$ as in the previous hybrid and then check if $y' = y$ and if for each $i \in H$, if $\sigma_i' = \sigma_i$. For every $i \in H$, such that the above check passes, we instruct the ideal functionality to deliver the output of P_i. For all the other parties, we instruct them to abort. In Lemma 9.6 we show that H_4 and H_4 are statistically indistinguishable from the security of the MAC scheme.

The proof ends with the observation that H_5 is identically distributed to the ideal world execution.

Lemma 9.2 (Indistinguishability of hybrids H_0 and H_1). *Let WL be a secure watchlist protocol, then the hybrids H_0 and H_1 are indistinguishable.*

Proof. We prove this lemma by contradiction. We assume an adversary $\mathcal{A}$ that successfully distinguishes between the hybrids H_0 and H_1 and use it to construct an adversary $\mathcal{A}'$ that breaks the list-simulateability of the underlying watchlist-protocol. The adversary $\mathcal{A}'$ behaves as follows:

The adversary $\mathcal{A}'$ acts as an augmented machine between the challenger of the underlying watchlist protocol wl and the adversary $\mathcal{A}$. In more detail, upon receiving the first round of the watchlist protocol $\{\mathsf{wl}_1^i\}_{i\in H}$, the adversary $\mathcal{A}'$ forwards it to the adversary to receive the first round of the watchlist protocol from the adversary $\{\mathsf{wl}_1^i\}_{i\in M}$. For all further queries, i.e. $\{\mathsf{wl}_1^r\}_{i\in H}$ for all $r \in \{2,3\}$, which also includes rewinds that are performed by the underlying challenger, the adversary $\mathcal{A}'$ computes the corresponding messages of the inner protocol Π_h honestly for all $h \in [m]$ and $i \in H$. Furthermore, during all these interactions with the underlying challenger, the reduction also learns the value that is used by the challenger as an input to generate its message. This value is used by the reduction to generate the other messages that require the input of the watchlist protocol.

In case that the underlying challenger generates the messages of the watchlist protocol honestly, the adversary $\mathcal{A}'$ simulates H_0 towards $\mathcal{A}$ and in the case that the underlying challenger simulates the messages of the watchlist protocol, then the adversary $\mathcal{A}'$ simulates H_1 towards $\mathcal{A}$. This concludes the proof of the lemma. □

Lemma 9.3 (Indistinguishability of hybrids H_1 and H_2). *The hybrids H_1 and H_2 are statistically indistinguishable.*

Proof. This proof works exactly in the same way as the proof of [IKSS21, Claim 6.7]. We recap it here verbatim for completeness. Fix any honest party P_i. Note that P_i aborts in H_2 if $|K_i \cap C'| \neq 0$. We show that if $|C'| > \lambda n^2$ then the probability of $|K_i \cap C'| = 0$ is negligible.

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

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

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

Lemma 9.4 (Indistinguishability of hybrids H_2 and H_3). *Let Π_h be a secure inner-protocol and Φ be a secure outer protocol then the hybrids H_2 and H_3 are indistinguishable.*

Proof. The proof follows from the observation that the security of our protocol Π_h described in Appendix A holds under parallel composition, and from the fact that $\mathcal{S}_{\mathsf{out}}$ satisfies Definition A.1. Indeed, due to the security of Φ we have that the following distributions are indistinguishable for every $\{z_i\}_{i\in H}$

$\{\{\phi_1^{i\to h}\}_{i\in H,h\in\mathsf{Cs}}, \{\phi_2^h\}_{h\in\mathsf{Us}} : \forall i \in H\ (\phi_1^{i\to 1},\ldots,\phi_1^{i\to m}) \leftarrow \Phi_1(1^\lambda, i, z_i), \forall h \in \mathsf{Us}, \phi_2^h \leftarrow \Phi_2(h,(\phi_1^{1\to h},\ldots,\phi_1^{n\to h}))\}$

$\{\{\phi_1^{i\to h}\}_{i\in H,h\in\mathsf{Cs}}, \{\phi_2^h\}_{h\in\mathsf{Us}} : \forall i \in H\ (\phi_1^{i\to 1},\ldots,\phi_1^{i\to m}) \leftarrow \Phi_1(1^\lambda, i, z_i), \forall h \in \mathsf{Us}, \phi_2^h \leftarrow \mathsf{Sim}_\Phi(\{\phi_1^{i\to h}\}_{i\in H,h\in\mathsf{Cs}}, \{\phi_1^{i\to h}\}_{i\in M,k\in\mathsf{Us}}\}$

condition on answering to a query of Sim_Φ of the form $\{z_i\}_{i\in M}$ as follows

1. For each $i \in [n]$ parse z_i as x_1, k_i.
2. Compute $y := f(x_1,\ldots,x_n)$.

3. Compute $\sigma_i := \mathsf{MAC}(\mathsf{k}_i, y)$ for all $i \in [n]$
4. Return $(y, \sigma_1, \ldots, \sigma_n)$.

□

Lemma 9.5 (Indistinguishability of hybrids H_2 and H_3). *If Φ is a secure outer protocol, then H_2 and H_3 are indistinguishable.*

Proof. We argue that if an adversary distinguishes between the two hybrids then we can contradict the fact that the first rounds generated from the honest parties are identically distributed, regardless of the input used to compute the messages of Φ_1. The proof works as follows. We fix a first round and second round of the execution executed accordingly to H_2 (and H_3). Note that by definition, the simulator of inner protocols does not need the input to compute the first round (which is sent in the second round). We define an auxiliary input, that contains all the information that can be extracted from the messages generated from the adversary in the first and in the second rounds. This in particular means that we can check whether the first round of i-th execution of the inner protocol has been computed correctly, for each $i \in [m]$. We can also extract the receiver's inputs from the watchlist protocol. We denote the indices of all the inner protocol executions that have been computed incorrectly by the adversary and the indices that are input by the adversary as part of the watchlist protocol executions (when the adversary acts as the receiver) with Bad, and the remaining indices with Good. We give this auxiliary input to our reduction, which works as follows.

1. If $|\mathsf{Bad}| \geq \lambda n^2$ then return a random bit and stop, else send to the challenger Bad, to denote the indices of the corrupted parties.
2. Upon receiving the shares $\{\phi_1^{i \to h}\}_{i \in H, h \in [\mathsf{Bad}]}$, instruct $\mathcal{D}^h$ to return the share $\{\phi_1^{i \to h}\}_{i \in H}$ whenever Sim_{Π_h} makes a query, for each $h \in \mathsf{Bad}$. Instruct $\mathcal{D}^h$ to return a random value whenever Sim_{Π_h} makes a query, for each $h \in \mathsf{Good}$.
3. Complete the third round of the protocol using the inner protocol simulators answering to their queries as described above, and by running the simulator of the watchlist protocol (note that this simulator can run in straight-line due to the information we have in the auxiliary input).
4. Upon receiving the third round of the protocol from the adversary, the watchlist simulator returns $\{r_{j,h}, \phi_1^{j \to h}\}_{j \in M, h \in [m]}$. If the number of invalid defenses received is greater or equal than λn^2 then return a random bit and stop, else send $\{r_{j,h}, \phi_1^{j \to h}\}_{j \in M, h \in [m]}$ to the challenger.
5. Upon receiving $\{\phi_2^h\}_{h \in \mathsf{Good}}$, use the ϕ_2^h to answer the query to the ideal world that Sim_{Π_h} makes, for each $h \in \mathsf{Good}$.
6. Complete the remaining steps of the protocol following H_2 (and H_3), and return whatever the adversary returns.

The proof ends with the observation that if the messages of the challenger have been simulated (i.e., computed with respect to a default input), then the adversary's behavior corresponds to the one he has in H_2, else it corresponds to H_2. □

Lemma 9.6 (Indistinguishability of hybrids H_3 and H_4). *Let MAC be a secure MAC scheme, then the hybrids H_3 and H_4 are indistinguishable.*

Proof. Therefore, the only difference between the two hybrids is that in H_4 it is checked that $y = y'$ and that for each $i \in H$ $\sigma_i' = \sigma_i$. For the honest parties where this check fails, they are instructed to abort. All other honest parties are instructed to output y. In hybrid H_4, on the other hand, the honest parties are instructed to do the MAC verification and, depending on the result, they abort or output y'. If an honest party does not abort in H_4, then the correctness of the verification procedure implies that it does not abort in H_3. Assume that there exists an honest party P_i that aborts in H_3, but that does not abort with non-negligible probability in H_5, then this means that $(y', \sigma_i') \neq (y, \sigma_i)$ and that the verification procedure on (y', σ_i') outputs 1. This contradicts the security of the MAC scheme and therefore concludes the proof. □

Theorem 9.1 further implies the following corollary.

Corollary 9.7. *Let* wl *be the watchlist protocol for the high min-entropy random variable* $\mathcal{X}$ *defined above, let* Π_h *be a secure inner-protocol,* Φ *a secure outer protocol and* MAC *a secure MAC scheme, then* Π *realizes any input-less functionality with black-box use of the primitives against any PPT adversaries.*

To argue that our corollary holds, we observe the following. A malicious generic PPT adversary is able to perform two types of attacks: it can learn the inputs of the honest parties and it can force wrong outputs on the honest parties. In the case of input-less functionalities, there is nothing to learn about the honest parties' inputs. Hence, we need to make sure that the adversary cannot force an incorrect output (i.e., the probability that an output occurs in the ideal world should be the same as the probability that the same output occurs in the real world up to a negligible factor). If an adversary forces the honest parties to produce a wrong output, then it must be that these honest parties have received an accepting transcript (i.e., no abort was triggered). Therefore, by Theorem 9.1 (and by the definition of sometimes aborting security), there exists a simulator, which directly results in the claim of Corollary 9.7. Note that this implication may not hold if we assume security against non-aborting adversaries (i.e., adversaries that abort with negligible probability only).

Acknowledgements

This work is supported in part by DARPA under Cooperative Agreement HR0011-20-2-0025, the Algorand Centers of Excellence programme managed by Algorand Foundation, NSF grants CNS-2246355, CCF-2220450 and CNS-2001096, US-Israel BSF grant 2015782, Amazon Faculty Award, Cisco Research Award and Sunday Group. Any views, opinions, findings, conclusions or recommendations contained herein are those of the author(s) and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, the Department of Defense, the Algorand Foundation, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

AAG$^+$16. D. Aggarwal, S. Agrawal, D. Gupta, H. K. Maji, O. Pandey, and M. Prabhakaran. Optimal computational split-state non-malleable codes. In *TCC 2016-A, Part II*, *LNCS* 9563, pages 393–417. Springer, Heidelberg, January 2016. (Page 12.)

ABG$^+$20. B. Applebaum, Z. Brakerski, S. Garg, Y. Ishai, and A. Srinivasan. Separating two-round secure computation from oblivious transfer. In *ITCS 2020*, pages 71:1–71:18. LIPIcs, January 2020. (Pages 86, 94, and 95.)

ACJ17. P. Ananth, A. R. Choudhuri, and A. Jain. A new approach to round-optimal secure multiparty computation. In *CRYPTO 2017, Part I*, *LNCS* 10401, pages 468–499. Springer, Heidelberg, August 2017. (Page 3.)

ADN$^+$19. D. Aggarwal, I. Damgård, J. B. Nielsen, M. Obremski, E. Purwanto, J. Ribeiro, and M. Simkin. Stronger leakage-resilient and non-malleable secret sharing schemes for general access structures. In *CRYPTO 2019, Part II*, *LNCS* 11693, pages 510–539. Springer, Heidelberg, August 2019. (Page 13.)

AIR01. W. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: How to sell digital goods. In *EUROCRYPT 2001*, *LNCS* 2045, pages 119–135. Springer, Heidelberg, May 2001. (Pages 3 and 12.)

AJL$^+$12. G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *EUROCRYPT 2012*, *LNCS* 7237, pages 483–501. Springer, Heidelberg, April 2012. (Page 15.)

BD18. Z. Brakerski and N. Döttling. Two-message statistically sender-private OT from LWE. In *TCC 2018, Part II*, *LNCS* 11240, pages 370–390. Springer, Heidelberg, November 2018. (Pages 3, 11, and 12.)

BGI$^+$18. E. Boyle, N. Gilboa, Y. Ishai, H. Lin, and S. Tessaro. Foundations of homomorphic secret sharing. In *ITCS 2018*, pages 21:1–21:21. LIPIcs, January 2018. (Pages 86, 94, and 95.)

BGJ$^+$18. S. Badrinarayanan, V. Goyal, A. Jain, Y. T. Kalai, D. Khurana, and A. Sahai. Promise zero knowledge and its applications to round optimal MPC. In *CRYPTO 2018, Part II*, *LNCS* 10992, pages 459–487. Springer, Heidelberg, August 2018. (Pages 3, 4, 5, and 9.)

BHP17. Z. Brakerski, S. Halevi, and A. Polychroniadou. Four round secure computation without setup. In *TCC 2017, Part I*, *LNCS* 10677, pages 645–677. Springer, Heidelberg, November 2017. (Page 3.)

BMR90. D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols (extended abstract). In *22nd ACM STOC*, pages 503–513. ACM Press, May 1990. (Pages 3 and 95.)

CCG+20. A. R. Choudhuri, M. Ciampi, V. Goyal, A. Jain, and R. Ostrovsky. Round optimal secure multiparty computation from minimal assumptions. In *TCC 2020, Part II*, *LNCS* 12551, pages 291–319. Springer, Heidelberg, November 2020. (Pages 3 and 19.)

CGL16. E. Chattopadhyay, V. Goyal, and X. Li. Non-malleable extractors and codes, with their many tampered extensions. In *48th ACM STOC*, pages 285–298. ACM Press, June 2016. (Page 13.)

COWZ22. M. Ciampi, R. Ostrovsky, H. Waldner, and V. Zikas. Round-optimal and communication-efficient multiparty computation. In *EUROCRYPT 2022, Part I*, *LNCS* 13275, pages 65–95. Springer, Heidelberg, May / June 2022. (Page 3.)

CRSW22. M. Ciampi, D. Ravi, L. Siniscalchi, and H. Waldner. Round-optimal multi-party computation with identifiable abort. In *EUROCRYPT 2022, Part I*, *LNCS* 13275, pages 335–364. Springer, Heidelberg, May / June 2022. (Page 3.)

GGH+13. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013. (Page 13.)

GIS18. S. Garg, Y. Ishai, and A. Srinivasan. Two-round MPC: Information-theoretic and black-box. In *TCC 2018, Part I*, *LNCS* 11239, pages 123–151. Springer, Heidelberg, November 2018. (Pages 86, 94, and 95.)

GJK15. V. Goyal, A. Jain, and D. Khurana. Witness signatures and non-malleable multi-prover zero-knowledge proofs. Cryptology ePrint Archive, Report 2015/1095, 2015. https://eprint.iacr.org/2015/1095. (Page 13.)

GK96. O. Goldreich and A. Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, June 1996. (Pages 23, 34, and 44.)

GKP17. S. Garg, S. Kiyoshima, and O. Pandey. On the exact round complexity of self-composable two-party computation. In *EUROCRYPT 2017, Part II*, *LNCS* 10211, pages 194–224. Springer, Heidelberg, April / May 2017. (Page 10.)

GMPP16. S. Garg, P. Mukherjee, O. Pandey, and A. Polychroniadou. The exact round complexity of secure computation. In *EUROCRYPT 2016, Part II*, *LNCS* 9666, pages 448–476. Springer, Heidelberg, May 2016. (Page 3.)

GMW87. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *19th ACM STOC*, pages 218–229. ACM Press, May 1987. (Page 3.)

Goy11. V. Goyal. Constant round non-malleable protocols using one way functions. In *43rd ACM STOC*, pages 695–704. ACM Press, June 2011. (Page 3.)

GPR16. V. Goyal, O. Pandey, and S. Richelson. Textbook non-malleable commitments. In *48th ACM STOC*, pages 1128–1141. ACM Press, June 2016. (Pages 10 and 11.)

GRRV14. V. Goyal, S. Richelson, A. Rosen, and M. Vald. An algebraic approach to non-malleability. In *55th FOCS*, pages 41–50. IEEE Computer Society Press, October 2014. (Page 10.)

GS18. S. Garg and A. Srinivasan. Two-round multiparty secure computation from minimal assumptions. In *EUROCRYPT 2018, Part II*, *LNCS* 10821, pages 468–499. Springer, Heidelberg, April / May 2018. (Page 88.)

GSZ20. V. Goyal, A. Srinivasan, and C. Zhu. Multi-source non-malleable extractors and applications. Cryptology ePrint Archive, Report 2020/157, 2020. https://eprint.iacr.org/2020/157. (Page 13.)

HHPV18. S. Halevi, C. Hazay, A. Polychroniadou, and M. Venkitasubramaniam. Round-optimal secure multi-party computation. In *CRYPTO 2018, Part II*, *LNCS* 10992, pages 488–520. Springer, Heidelberg, August 2018. (Pages 3 and 10.)

HHPV21. S. Halevi, C. Hazay, A. Polychroniadou, and M. Venkitasubramaniam. Round-optimal secure multi-party computation. *Journal of Cryptology*, 34(3):19, July 2021. (Page 7.)

HK12. S. Halevi and Y. T. Kalai. Smooth projective hashing and two-message oblivious transfer. *Journal of Cryptology*, 25(1):158–193, January 2012. (Page 12.)

HKN+05. D. Harnik, J. Kilian, M. Naor, O. Reingold, and A. Rosen. On robust combiners for oblivious transfer and other primitives. In *EUROCRYPT 2005*, *LNCS* 3494, pages 96–113. Springer, Heidelberg, May 2005. (Page 4.)

IKP10. Y. Ishai, E. Kushilevitz, and A. Paskin. Secure multiparty computation with minimal interaction. In *CRYPTO 2010*, *LNCS* 6223, pages 577–594. Springer, Heidelberg, August 2010. (Pages 14, 15, and 76.)

IKSS21. Y. Ishai, D. Khurana, A. Sahai, and A. Srinivasan. On the round complexity of black-box secure MPC. In *CRYPTO 2021, Part II*, *LNCS* 12826, pages 214–243, Virtual Event, August 2021. Springer, Heidelberg. (Pages 3, 4, 5, 6, 7, 8, 11, 12, 13, 15, 17, 48, 50, 53, 63, 65, 74, 76, and 81.)

IKSS22. Y. Ishai, D. Khurana, A. Sahai, and A. Srinivasan. Round-optimal black-box protocol compilers. In *EUROCRYPT 2022, Part I*, *LNCS* 13275, pages 210–240. Springer, Heidelberg, May / June 2022. (Page 87.)

IKSS23. Y. Ishai, D. Khurana, A. Sahai, and A. Srinivasan. Round-optimal black-box mpc in the plain model. Cryptology ePrint Archive, Paper 2023/1173, 2023. `https://eprint.iacr.org/2023/1173`. (Page 3.)

IPS08. Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO 2008*, *LNCS* 5157, pages 572–591. Springer, Heidelberg, August 2008. (Pages 3, 5, 8, and 76.)

Kal05. Y. T. Kalai. Smooth projective hashing and two-message oblivious transfer. In *EUROCRYPT 2005*, *LNCS* 3494, pages 78–95. Springer, Heidelberg, May 2005. (Pages 3 and 12.)

Kil88. J. Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988. (Page 3.)

KO04. J. Katz and R. Ostrovsky. Round-optimal secure two-party computation. In *CRYPTO 2004*, *LNCS* 3152, pages 335–354. Springer, Heidelberg, August 2004. (Page 3.)

KOS03. J. Katz, R. Ostrovsky, and A. Smith. Round efficiency of multi-party computation with a dishonest majority. In *EUROCRYPT 2003*, *LNCS* 2656, pages 578–595. Springer, Heidelberg, May 2003. (Page 3.)

KOS18. D. Khurana, R. Ostrovsky, and A. Srinivasan. Round optimal black-box “commit-and-prove”. In *TCC 2018, Part I*, *LNCS* 11239, pages 286–313. Springer, Heidelberg, November 2018. (Page 10.)

Lin16. Y. Lindell. How to simulate it - A tutorial on the simulation proof technique. Cryptology ePrint Archive, Report 2016/046, 2016. `https://eprint.iacr.org/2016/046`. (Pages 23, 24, 34, 35, and 44.)

MOSV22. V. Madathil, C. Orsini, A. Scafuro, and D. Venturi. From privacy-only to simulatable OT: black-box, round-optimal, information-theoretic. In *3rd Conference on Information-Theoretic Cryptography, ITC 2022, July 5-7, 2022, Cambridge, MA, USA*, *LIPIcs* 230, pages 5:1–5:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. (Page 9.)

NP01. M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *12th SODA*, pages 448–457. ACM-SIAM, January 2001. (Page 3.)

Ode09. G. Oded. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, USA, 1st edition, 2009. (Pages 13 and 14.)

Pas04. R. Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *36th ACM STOC*, pages 232–241. ACM Press, June 2004. (Page 3.)

PS21. A. Patra and A. Srinivasan. Three-round secure multiparty computation from black-box two-round oblivious transfer. In *CRYPTO 2021, Part II*, *LNCS* 12826, pages 185–213, Virtual Event, August 2021. Springer, Heidelberg. (Page 88.)

PW10. R. Pass and H. Wee. Constant-round non-malleable commitments from sub-exponential one-way functions. In *EUROCRYPT 2010*, *LNCS* 6110, pages 638–655. Springer, Heidelberg, May / June 2010. (Page 3.)

SSR08. B. Shankar, K. Srinathan, and C. P. Rangan. Alternative protocols for generalized oblivious transfer. In *Distributed Computing and Networking*, pages 304–309, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. (Pages 5 and 40.)

Wee10. H. Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *51st FOCS*, pages 531–540. IEEE Computer Society Press, October 2010. (Page 3.)

Yao86. A. C.-C. Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986. (Page 3.)

A Inner Protocol

In this section, we present our three-round semi-honest MPC protocol Π_h with black box uses of semi-honest oblivious transfer, in the dishonest majority setting, used for the protocol described in Section 9.

We start by introducing the formal security notion that our protocol achieves in Appendix A.1 and then continue by presenting a protocol for the 3MULTPlus functionality in Appendix A.2. The 3MULTPlus takes as inputs (x_1, y_1), (x_2, y_2) and (x_3, y_3) respectively from parties P_1, P_2 and P_3 and delivers to all the parties (and not only to P_1, P_2 and P_3) $x_1 \cdot x_2 \cdot x_3 + y_1 + y_2 + y_3$.

In Appendix A.3, we then argue that a protocol for the 3MULTPlus functionality is sufficient to obtain a three-round protocol for general functionalities by applying the round-preserving transformations of [BGI+18, GIS18, ABG+20].

A.1 Definition

In this section, we specify the security requirements that we require from the inner-protocol Π_h. Informally, these are:

Semi-malicious Security: Suppose that at the end of the second round an adversary $\mathcal{A}$ produces input and randomness that explains the messages sent by all the corrupted parties in the previous rounds, then the last round message sent by the honest parties reveals no other information about their inputs except what is leaked from the output of the function. Here, the output of the function is generated by sampling the input on behalf of the honest parties from the distribution $\mathcal{D}$. We call this property *semi-malicious security.*

Honest Behavior: Assume that at the end of the second round, an adversary $\mathcal{A}$ is unable to produce input and randomness that explain the messages sent by the corrupted parties in the previous rounds, then the input of the honest parties are leacked to the adversary. We call this property *honest behavior.*

Now, we present our definition more formally:

Syntax. The third-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 or Π in the first r rounds. At the end of the interaction, parties run $\mathsf{out}_\Pi(\pi(3))$ to compute the output.[16]

Definition A.1. *We say that the protocol Π is an inner protocol for computing a function f with respect to the distribution $\mathcal{D}$ if it satisfies the following properties:*

Correctness: *We say that the protocol Π correctly computes a function f if for every choice of inputs x_i for party P_i,*

$$\Pr[\mathsf{out}_\Pi(\pi(3)) = f(x_1, \dots, x_n)] = 1,$$

where $\pi(3)$ denotes the transcript of the protocol Π when the input of P_i is x_i.

Security: *We capture all the security properties in a real/ideal security game. Let $\mathcal{A}$ be an adversary corrupting a subset of the parties indexed by the set M, let H be the set of indices denoting the honest parties and let $\mathcal{S}_{\mathsf{out}}$ be any stateful PPT algorithm such that for every $\{x_i\}_{i\in M}$ $\{y : \{x_i\}_{i\in H} \leftarrow \mathcal{D}, \{x_i\}_{i\in M}, y := f(x_1, \dots, x_n)\} \approx \{y : \{x'_i\}_{i\in M} \leftarrow \mathcal{S}_{\mathsf{out}}(\{x_i\}_{i\in M}), y' := f(\{x_i, x'_j\}_{i\in H, j\in M}), y \leftarrow \mathcal{S}_{\mathsf{out}}(y')\}$, the we require the existence of a simulator Sim_{Π_h} such that,*

$$\mathrm{Real}(\mathcal{A}, \mathcal{D}, \{r_i\}_{i\in H}) \approx_c \mathrm{Ideal}(\mathcal{A}, \mathsf{Sim}_{\Pi_h}, \mathcal{D}, \mathcal{S}_{\mathsf{out}}),$$

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

where the real and ideal experiments are described in the figure below and for $i \in H$, r_i is uniformly chosen and $\{x_i\}_{i \in M}$ are the input of the corrupted parties sent as part of their defence in the semi-malicious case. Sim_{Π_h} interacts with the ideal functionality, which takes as an input the inputs of the adversary $\{x_i\}_{i \in M}$, applies $\mathcal{S}_{\mathsf{out}}(\{x_i\}_{i \in M})$ to obtain $\{x'_i\}_{i \in M}$, samples the inputs on behalf of the honest parties $\{x_i\}_{i \in H}$, computes $y' := f(\{x_i, x'_j\}_{i \in H, j \in M})$, then computes $y \leftarrow \mathcal{S}_{\mathsf{out}}(y')$ and replies to the simulator Sim_{Π_h} with y.

Figure A.1: Security Game for the Inner Protocol

```
                         $\underline{\mathrm{Real}(\mathcal{A}, \mathcal{D}, \{r_i\}_{i \in H})}$
1. For each $i \in H$ sample $x_i \leftarrow \mathcal{D}$
2. For each $r \in [2]$:
   (a) For each $i \in H$, compute $\pi_r^i := \Pi_r(1^\lambda, i, x_i, \pi(r-1); r_i)$ where $\pi(0)$ is the null string.
   (b) Send $\{\pi_r^i\}_{i \in H}$ to $\mathcal{A}$.
   (c) Receive $\{\pi_r^i\}_{i \in M}$ from $\mathcal{A}$. If $r = 2$, receive $\{\pi_r^i, (x_i, r_i)\}_{i \in M}$ from $\mathcal{A}$.
3. Check if the messages sent by the corrupted parties in $\pi(2)$ are consistent with $\{x_i, r_i\}_{i \in M}$.
4. **Semi-Malicious Security:** If they are consistent:
   (a) For each $i \in H$, compute $\pi_3^i := \Pi_3(1^\lambda, i, x_i, \pi(2); r_i)$
   (b) Compute $y := f(x_1, \ldots, x_n)$.
   (c) Send $(\{\pi_3^i\}_{i \in H}, y)$ to $\mathcal{A}$.
5. **Honest Behavior:** If they are not consistent:
   (a) For each $i \in H$, compute $\pi_3^i := \Pi_3(1^\lambda, i, x_i, \pi(2); r_i)$
   (b) Sent $\{\pi_3^i\}_{i \in H}$ to $\mathcal{A}$.
6. Receive $\{\pi_3^i\}_{i \in M}$ from $\mathcal{A}$.
7. Output the view of $\mathcal{A}$ and $\mathsf{out}_\Pi(\pi(3))$.

                         $\underline{\mathrm{Ideal}(\mathcal{A}, \mathsf{Sim}_\Pi, \mathcal{D}, \mathcal{S}_{\mathsf{out}})}$
1. For each $i \in H$, compute $\pi_1^i := \mathsf{Sim}_\Pi(1^\lambda, i, 0^\lambda)$.
2. Send $\{\pi_1^i\}_{i \in H}$ to $\mathcal{A}$ and receive $\{\pi_1^i\}_{i \in M}$ from $\mathcal{A}$.
3. For each $i \in H$, compute $(\pi_2^i, \{\tilde{x}_i, \tilde{r}_i\}_{i \in H}) := \mathsf{Sim}_\Pi^{\mathcal{D}}(1^\lambda, i, \pi(1))$.
4. Send $\{\pi_2^i\}_{i \in H}$ to $\mathcal{A}$ and receive $\{\pi_2^i, (x_i, r_i)\}_{i \in M}$ from $\mathcal{A}$.
5. Check if the messages sent by the corrupted parties in $\pi(2)$ are consistent with $\{x_i, r_i\}_{i \in M}$.
6. **Semi-Malicious Security:** If they are consistent:
   (a) Query the ideal functionality using $\{x_i\}_{i \in M}$ to receive $y$.
   (b) For each $i \in H$, compute $\pi_3^i := \mathsf{Sim}_\Pi(1^\lambda, i, y, \{x_i, r_i\}_{i \in M}, \pi(2))$.
   (c) Send $(\{\pi_3^i\}_{i \in H}, y)$ to $\mathcal{A}$.
7. **Honest Behavior:** If they are not consistent:
   (a) For each $i \in H$, compute $\pi_3^i := \mathsf{Sim}_\Pi^{\mathcal{D}}(1^\lambda, i, \pi(2))$.
   (b) Send $\{\pi_3^i\}_{i \in H}$ to $\mathcal{A}$.
8. Receive $\{\pi_3^i\}_{i \in M}$ from $\mathcal{A}$.
9. Output the view of $\mathcal{A}$ and $\mathsf{out}_\Pi(\pi(3))$.
```

A.2 Construction of the 3MULTPlus protocol

Before presenting how to construct the protocol for the 3MULTPlus functionality, we introduce the security requirements that we require from the underlying OT protocol. These security requirements are the same as the ones in [IKSS22, Appendix B.1] and, for its realization, we can rely on the protocol presented in [IKSS22, Appendix A.1].

Syntax. Let $\mathsf{OT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{out}_{\mathsf{OT}})$ be a two-round oblivious transfer protocol. The OT_1 algorithm takes in the security parameter 1^λ and the receiver's choice bit b and outputs the first round message ot_1

along with a secret key sk. The OT_2 algorithm takes in the first round message ot_1, the sender inputs m_0, m_1 and outputs the sender message ot_2. The $\mathsf{out}_{\mathsf{OT}}$ algorithm takes in the sender message ot_2 and the secret key sk and outputs the message m_b. We say that the OT protocol is a two-round oblivious transfer with equivocal receiver security [GS18, PS21] if it satisfies the following properties:

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

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

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

Equivocal Receiver Security: There exists a special algorithm $\mathsf{Sim}^{Eq}_{\mathsf{OT}}$ that on input 1^λ outputs $(\mathsf{ot}_1, \mathsf{sk}_0, \mathsf{sk}_1)$ such that for any $b \in \{0, 1\}$,

$$\{(\mathsf{ot}_1, \mathsf{sk}_b) : (\mathsf{ot}_1, \mathsf{sk}_0, \mathsf{sk}_1) \leftarrow \mathsf{Sim}^{Eq}_{\mathsf{OT}}(1^\lambda)\} \approx_c \{(\mathsf{ot}_1, \mathsf{sk}) : (\mathsf{ot}_1, \mathsf{sk}) \leftarrow \mathsf{OT}_1(1^\lambda, b)\}$$

Weak Adaptive Semi-Honest Sender Security: There exists a (stateful) simulator Sim_S such that for every (stateful) adversary $\mathcal{A}$ corrupting the receiver and any sender inputs (m_0, m_1) (such that $|m_0| = |m_1|$), we have:

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

where the distributions Real_S and Ideal_S are described in Fig. A.2.

Figure A.2: Real and ideal world for sender security

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

1. Sample uniform random tapes r for the receiver and s for the sender.
2. Send r to $\mathcal{A}$.
3. $\mathcal{A}$ outputs the receiver input b.
4. Generate the message msg_1 using the input and the random tape sampled above.
5. Run $\mathcal{A}(\mathsf{msg}_1)$.
6. If $\mathcal{A}$ outputs a special symbol $\mathsf{corrupt}$, then send the random tape of the sender to $\mathcal{A}$ and output the view of $\mathcal{A}$. Otherwise, proceed to the next step.
7. Generate the last round message msg_2 as the honest sender and send it to $\mathcal{A}$.
8. Output the view of $\mathcal{A}$.

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

1. Sample uniform random tape of the receiver r.
2. Send r to $\mathcal{A}$.
3. $\mathcal{A}$ outputs the receiver input b.
4. Use the random tape r and the input b to generate msg_1.
5. Run $\mathcal{A}(\mathsf{msg}_1)$.
6. If $\mathcal{A}$ outputs a special symbol $\mathsf{corrupt}$, then run the simulator on (m_0, m_1) to obtain the random tape of the sender. Send this to $\mathcal{A}$ and output its view. Otherwise, proceed to the next step.
7. Generate the last round message msg_2 using $\mathsf{Sim}_S(b, r, m_b)$ and send it to $\mathcal{A}$.
8. Output the view of $\mathcal{A}$.

Now, using the OT protocol descried above, we are ready to present our protocol for the $\mathsf{3MULTPlus}$ functionality, which follows the template of [PS21].

Figure A.3: $\mathsf{3MULTPlus}$ of [PS21]

Initialization: Each party P_i uses (x_i, y_i) for $i \in [3]$ as its input in the protocol below.

Round 1:
- P_1 computes $(\mathsf{ot}_1, \mathsf{sk}) \leftarrow \mathsf{OT}_1(1^\lambda, x_1)$.
- P_2 chooses random bits $x_{2,0}, x_{2,1} \leftarrow \{0, 1\}$ subject to $x_2 = x_{2,0} + x_{2,1}$ and computes $(\mathsf{ot}^0_1, \mathsf{sk}_0) \leftarrow \mathsf{OT}_1(1^\lambda, x_{2,0})$ and $(\mathsf{ot}^1_1, \mathsf{sk}_1) \leftarrow \mathsf{OT}_1(1^\lambda, x_{2,1})$.
- P_3 computes $(\mathsf{ot}^3_1, \mathsf{sk}_3) \leftarrow \mathsf{OT}_1(1^\lambda, x_3)$.

- P_1 broadcasts ot_1, P_2 broadcasts $(\mathsf{ot}_1^0, \mathsf{ot}_1^1)$ and P_3 broadcasts ot_1^3.
- For every $i \in [3]$, P_i chooses a random additive secret sharing of 0 given by $(\delta_1^i, \delta_2^i, \delta_3^i)$ and sends the share δ_j^i to party P_j for $j \in [3] \setminus \{i\}$ via private channels. We note that we can simulate a single round of private channel messages in two-rounds over public channels by making use of a two-round oblivious transfer.

Round 2:
- P_2 computes $\mathsf{ot}_2 \leftarrow \mathsf{OT}_2(\mathsf{ot}_1, (x_{2,0}, \mathsf{sk}_0), (x_{2,1}, \mathsf{sk}_1))$. It then chooses random bits $x_{2,0,0}, x_{2,0,1} \leftarrow \{0,1\}$ subject to $x_{2,0} = x_{2,0,0} + x_{2,0,1}$. It computes $\mathsf{ot}_2^3 \leftarrow \mathsf{OT}_2(\mathsf{ot}_1^3, x_{2,0,0}, x_{2,0,1})$.
- P_3 chooses random bits $x_{3,0}, x_{3,1} \leftarrow \{0,1\}$ subject to $x_3 = x_{3,0} + x_{3,1}$. For each $b \in \{0,1\}$, it first computes $\mathsf{ot}_2^b \leftarrow \mathsf{OT}_2(\mathsf{ot}_1^b, x_{3,0}, x_{3,1})$. It then computes $\overline{\mathsf{ot}_2} \leftarrow \mathsf{OT}_2(\mathsf{ot}_1, \mathsf{ot}_2^0, \mathsf{ot}_2^1)$.
- P_2 sends ot_2 to P_1 via private channel and ot_2^3 to P_3 via private channel. P_3 sends $\overline{\mathsf{ot}_2}$ to P_1 via private channel.

Round 3:
- For each $i \in [3]$, P_i computes $\delta_i = \sum_{j \in [3]} \delta_i^j$.
- P_2 sets $z_2 := x_{2,0,0} + y_2 + \delta_2$.
- P_3 computes $x_{2,0,x_3} := \mathsf{out}_{\mathsf{OT}}(\mathsf{ot}_2^3, (x_3, \mathsf{sk}_3))$ and sets $z_3 := x_{2,0,x_3} + x_{3,0} + y_3 + \delta_3$.
- P_1 computes $(x_{2,x_1}, \mathsf{sk}_{x_1}) := \mathsf{out}_{\mathsf{OT}}(\mathsf{ot}_1, (x_1, \mathsf{sk}))$ and $\mathsf{ot}_2^{x_1} := \mathsf{out}_{\mathsf{OT}}(\overline{\mathsf{ot}_2}, (x_1, \mathsf{sk}))$. It then computes $x_{3,x_{2,x_1}} := \mathsf{out}_{\mathsf{OT}}(\mathsf{ot}_2^{x_1}, (x_{2,x_1}, \mathsf{sk}_{x_1}))$. It then sets $z_1 := x_{3,x_{2,x_1}} + y_1 + \delta_1$.
- P_1 broadcasts z_1, P_2 broadcasts z_2 and P_3 broadcasts z_3.

Output: Every party outputs $z_1 + z_2 + z_3$.

Theorem A.2. *Assuming the existence of equivocal oblivious transfer (see Definition Section 2.5), then the* $\mathsf{3MULTPlus}$ *protocol described in Figure A.3 satisfies Definition A.1.*

Proof. To prove this theorem, we first describe a simulator in Figure A.4 below.

Figure A.4: Simualtor Sim of $\mathsf{3MULTPlus}$

Round 1:
- To generate the first round messages of the honest parties, Sim does the following:
  1. If $P_1 \in H$, then Sim computes $(\mathsf{ot}_1, \mathsf{sk}_0', \mathsf{sk}_1') \leftarrow \mathsf{Sim}_{\mathsf{OT}}^{Eq}(1^\lambda)$.
  2. If $P_2 \in H$, then for each $b \in \{0,1\}$, Sim computes $(\mathsf{ot}_1^b, \mathsf{sk}_{b,0}', \mathsf{sk}_{b,1}') \leftarrow \mathsf{Sim}_{\mathsf{OT}}^{Eq}(1^\lambda)$.
  3. If $P_3 \in H$, then Sim computes $(\mathsf{ot}_1^3, \mathsf{sk}_0'', \mathsf{sk}_1'') \leftarrow \mathsf{Sim}_{\mathsf{OT}}^{Eq}(1^\lambda)$.
  4. It sends the above computed messages on behalf of the honest parties to the adversary.

Round 2: On receiving the first round messages from $\mathcal{A}$, the distribution $\mathcal{D}$, Sim proceeds as follows:
1. If $P_2 \in H$, sample $(x_2, y_2) \leftarrow \mathcal{D}$ and compute $\mathsf{ot}_2 \leftarrow \mathsf{OT}_2(\mathsf{ot}_1, (x_{2,0}, \mathsf{sk}_0), (x_{2,1}, \mathsf{sk}_1))$ with $x_{2,0}, x_{2,1} \leftarrow \{0,1\}$ subject to $x_2 = x_{2,0} + x_{2,1}$ and $\mathsf{sk}_0 = \mathsf{sk}_{0,x_{2,0}}'$, $\mathsf{sk}_1 = \mathsf{sk}_{1,x_{2,1}}'$. Similarly, compute $\mathsf{ot}_2^3 \leftarrow \mathsf{OT}_2(\mathsf{ot}_1^3, x_{2,0,0}, x_{2,0,1})$, where $x_{2,0,0}, x_{2,0,1} \leftarrow \{0,1\}$ subject to $x_{2,0} = x_{2,0,0} + x_{2,0,1}$.
2. If $P_3 \in H$, sample $(x_3, y_3) \leftarrow \mathcal{D}$ and compute $\mathsf{ot}_2^b \leftarrow \mathsf{OT}_2(\mathsf{ot}_1^b, x_{3,0}, x_{3,1})$ for each $b \in \{0,1\}$ with $x_{3,0}, x_{3,1} \leftarrow \{0,1\}$ subject to $x_3 = x_{3,0} + x_{3,1}$. Afterwards, compute $\overline{\mathsf{ot}_2} \leftarrow \mathsf{OT}_2(\mathsf{ot}_1, \mathsf{ot}_2^0, \mathsf{ot}_2^1)$.

Send the computed messages to $\mathcal{A}$ along with inputs $\{(x_i, y_i)\}_{i \in H}$.

Round 3: On receiving the second round messages inputs and random tapes $\{(x_i, y_i, r_i)\}_{i \in M}$ from $\mathcal{A}$ we distinguish between two cases:

(a) At least one of the messages is inconsistent w.r.t. the inputs and random tapes provided by $\mathcal{A}$:
- Act for the honest parties as follows:

  If $P_1 \in H$:
  1. Sample $(x_1, y_1) \leftarrow \mathcal{D}$.
  2. Compute $\delta_1 = \sum_{j \in [3]} \delta_1^j$.
  3. Set $\mathsf{sk} := \mathsf{sk}_{x_1}'$.

4. Compute $(x_{2,x_1}, \mathsf{sk}_{x_1}) \coloneqq \mathsf{out}_{\mathsf{OT}}(\mathsf{ot}_1, (x_1, \mathsf{sk}))$ and $\mathsf{ot}_2^{x_1} \coloneqq \mathsf{out}_{\mathsf{OT}}(\overline{\mathsf{ot}_2}, (x_1, \mathsf{sk}))$. Afterwards, compute $x_{3,x_{2,x_1}} \coloneqq \mathsf{out}_{\mathsf{OT}}(\mathsf{ot}_2^{x_1}, (x_{2,x_1}, \mathsf{sk}_{x_1}))$ and set $z_1 \coloneqq x_{3,x_{2,x_1}} + y_1 + \delta_1$.
5. Broadcast z_1.

If $P_2 \in H$**:**

1. Compute $\delta_2 = \sum_{j\in[3]} \delta_2^j$.
2. Set $z_2 \coloneqq x_{2,0,0} + y_2 + \delta_2$.
3. Broadcast z_2.

If $P_3 \in H$**:**

1. Compute $\delta_3 = \sum_{j\in[3]} \delta_3^j$.
2. Set $\mathsf{sk}_3 \coloneqq \mathsf{sk}''_{x_3}$.
3. Compute $x_{2,0,x_3} \coloneqq \mathsf{out}_{\mathsf{OT}}(\mathsf{ot}_2^3, (x_3, \mathsf{sk}_3))$ and set $z_3 \coloneqq x_{2,0,x_3} + x_{3,0} + y_3 + \delta_3$.
4. Broadcast z_3.

(b) All the messages are consistent w.r.t. the inputs and random tapes provided by $\mathcal{A}$:

– In this case, Sim queries the ideal functionality using $\{(x_i, y_i)\}_{i\in M}$ to obtain z. Then Sim computes $\{z_i\}_{i\in M}$ using the transcript and the random tape of the adversary as follows:

If $P_1 \in M$**:**

1. Compute $\delta_1 = \sum_{j\in[3]} \delta_1^j$.
2. Compute $(x_{2,x_1}, \mathsf{sk}_{x_1}) \coloneqq \mathsf{out}_{\mathsf{OT}}(\mathsf{ot}_1, (x_1, \mathsf{sk}_1))$ and $\mathsf{ot}_2^{x_1} \coloneqq \mathsf{out}_{\mathsf{OT}}(\overline{\mathsf{ot}_2}, (x_1, \mathsf{sk}_1))$, where sk_1 can be derived from the random tape of the adversary. Afterwards, compute $x_{3,x_{2,x_1}} \coloneqq \mathsf{out}_{\mathsf{OT}}(\mathsf{ot}_2^{x_1}, (x_{2,x_1}, \mathsf{sk}_{x_1}))$ and set $z_1 \coloneqq x_{3,x_{2,x_1}} + y_1 + \delta_1$.

If $P_2 \in M$**:**

1. Compute $\delta_2 = \sum_{j\in[3]} \delta_2^j$.
2. Set $z_2 \coloneqq x_{2,0,0} + y_2 + \delta_2$.

If $P_3 \in M$**:**

1. Compute $\delta_3 = \sum_{j\in[3]} \delta_3^j$.
2. Compute $x_{2,0,x_3} \coloneqq \mathsf{out}_{\mathsf{OT}}(\mathsf{ot}_2^3, (x_3, \mathsf{sk}_3))$ and set $z_3 \coloneqq x_{2,0,x_3} + x_{3,0} + y_3 + \delta_3$. Here, sk_3 is computed using the random tape of the adversary.

Afterwards, choose $\{z_i\}_{i\in H}$ uniformly such that $\bigoplus_{i\in H} z_i = z \oplus \bigoplus_{i\in M} z_i$ and send $\{z_i\}_{i\in H}$ to $\mathcal{A}$.

Running time of the simulator. Since this simulator does not perform any rewinds, it clearly runs in polynomial time.

To prove the theorem and the indistinguihsability between the real and the ideal world, we proceed using a few hybrids. These hybrids are only interesting in the case that the adversary is consistent. In the case that the adversary is inconsistent there is nothing that we need to prove. Therefore, the following hybrids are used to argue security in the cast that the adversary is consistent and no corrupt command was issued.

Hybrid H_0: This hybrid corresponds to the real world.

Hybrid H_1: If $P_1 \notin H$, then this hybrid is skipped. Otherwise, compute $(\mathsf{ot}_1, \mathsf{sk}'_0, \mathsf{sk}'_1) \leftarrow \mathsf{Sim}_{\mathsf{OT}}^{Eq}(1^\lambda)$, set $\mathsf{sk} = \mathsf{sk}'_{x_1}$ with $(x_1, y_1) \leftarrow \mathcal{D}$ and proceed as in the honest protocol execution. This hybrid is computationally indistinguishable to the hybrid H_0, which can be shown by relying on the equivocal receiver security of the OT protocol. We prove this transition more formally in Lemma A.3.

Hybrid H_2: If $P_3 \notin H$, then this hybrid is skipped. Otherwise, compute $(\mathsf{ot}_1^3, \mathsf{sk}'_0, \mathsf{sk}'_1) \leftarrow \mathsf{Sim}_{\mathsf{OT}}^{Eq}(1^\lambda)$, set $\mathsf{sk}_3 = \mathsf{sk}'_{x_3}$ with $(x_3, y_3) \leftarrow \mathcal{D}$ and proceed as in the honest protocol execution. This hybrid is computationally indistinguishable to the previous hybrid, which can be shown by relying on the equivocal receiver security of the OT protocol. The proof of the hybrid transition here is similar to the proof of the hybrid transition from H_0 to H_1, therefore, we refer to this proof (Lemma A.3) for further details.

Hybrid H_3: If $P_2 \notin H$, then this hybrid is skipped. Otherwise, for each $b \in \{0,1\}$, compute $(\mathsf{sk}'_{b,0}, \mathsf{sk}'_{b,1}) \leftarrow \mathsf{Sim}_{\mathsf{OT}}^{Eq}(1^\lambda)$, set $\mathsf{sk}_0 = \mathsf{sk}'_{0,x_{2,0}}$ and $\mathsf{sk}_1 = \mathsf{sk}'_{1,x_{2,1}}$ with $(x_2, y_2) \leftarrow \mathcal{D}$ and $x_{2,0}, x_{2,1} \leftarrow \{0,1\}$ such that $x_2 = x_{2,0} + x_{2,1}$. Afterwards, proceed as in the protocol execution. This hybrid is computationally indistinguishable to the previous hybrid, which can be shown by relying on the equivocal receiver security of the OT protocol. The proof of the hybrid transition here is similar to the proof of the hybrid transition from H_1 to H_2 and H_0 to H_1, therefore, we refer to the proof of the transition from H_0 to H_1 (Lemma A.3) for further details.

After this hybrid, we distinguish between two cases: first, the messages are inconsistent w.r.t. the inputs and random tapes provided by $\mathcal{A}$ and, second, the messages are consistent w.r.t. the inputs and random tapes provided by $\mathcal{A}$. In the first, case, we simply rely on the equivocality of the OT protocol and behave as described in the description of the simulator (Figure A.4) for the honest parties. In more detail, we behave as in the honest execution of the protocol for the remaining rounds. In this case, the proof concludes here. In the second case, the next hybrids after receiving the first round from $\mathcal{A}$ run in exponential time to extract the inputs and randomness used by the adversary in the first round. The hybrids are defined as follows:

Hybrid H_4: If $P_2 \in M$ or $P_1 \in H$, this hybrid is skipped. Otherwise, ot_2 is generated as $\mathsf{ot}_2 \leftarrow \mathsf{OT}_2(\mathsf{ot}_1, (x_{2,x_1}, \mathsf{sk}_{x_1}), (x_{2,x_1}, \mathsf{sk}_{x_1}))$ instead of $\mathsf{ot}_2 \leftarrow \mathsf{OT}_2(\mathsf{ot}_1, (x_{2,0}, \mathsf{sk}_0), (x_{2,1}, \mathsf{sk}_1))$. This hybrid is computationally indistinguishable to the hybrid H_3, which can be shown by relying on the weak adaptive sender security of the OT protocol. We prove this transition more formally in Lemma A.4.

Hybrid H_5: If $P_2 \in M$ or $P_3 \in H$, this hybrid is skipped. Otherwise, ot_2^3 is generated as $\mathsf{ot}_2^3 \leftarrow \mathsf{OT}_2(\mathsf{ot}_1^3, x_{2,0,x_3}, x_{2,0,x_3})$ instead of $\mathsf{ot}_2^3 \leftarrow \mathsf{OT}_2(\mathsf{ot}_1^3, x_{2,0,0}, x_{2,0,1})$. This hybrid is computationally indistinguishable to the previous hybrid, which can be shown by relying on he weak adaptive sender security of the OT protocol. The proof of the hybrid transition here is similar to the proof of the hybrid transition from H_3 to H_4, therefore, we refer to this proof (Lemma A.4) for further details.

Hybrid H_6: If $P_3 \in M$ or $P_2 \in H$, this hybrid is skipped. Otherwise, ot_2^b is generated as $\mathsf{ot}_2^b \leftarrow \mathsf{OT}_2(\mathsf{ot}_1^b, x_{3,x_{2,b}}, x_{3,x_{2,b}})$ instead of $\mathsf{ot}_2^b \leftarrow \mathsf{OT}_2(\mathsf{ot}_1^b, x_{3,0}, x_{3,1})$ for each $b \in \{0,1\}$. This hybrid is computationally indistinguishable to the previous hybrid, which can be shown by relying on the the weak adaptive sender security of the OT protocol. The proof of the hybrid transition here is similar to the proof of the hybrid transition from H_3 to H_4, therefore, we refer to this proof (Lemma A.4) for further details.

Hybrid H_7: If $P_3 \in M$ or $P_1 \in H$, this hybrid is skipped. Otherwise, $\overline{\mathsf{ot}_2}$ is generated as $\overline{\mathsf{ot}_2} \leftarrow \mathsf{OT}_2(\mathsf{ot}_1, \mathsf{ot}_2^{x_1}, \mathsf{ot}_2^{x_1})$ instead of $\overline{\mathsf{ot}_2} \leftarrow \mathsf{OT}_2(\mathsf{ot}_1, \mathsf{ot}_2^0, \mathsf{ot}_2^1)$. This hybrid is computationally indistinguishable to the previous hybrid, which can be shown by relying on the the weak adaptive sender security of the OT protocol. The proof of the hybrid transition here is similar to the proof of the hybrid transition from H_3 to H_4, therefore, we refer to this proof (Lemma A.4) for further details.

Hybrid H_8: Let i^* be the smallest integer in H. In this hybrid, the simulator queries the ideal functionality using the inputs of the malicious parties and obtains z as a reply from the ideal functionality. Afterwards, it sets $z_{i^*} := z - \sum_{j \in [3] \setminus \{i^*\}} z_j$ instead of computing it as in the previous hybrid. In the previous hybrid z_{i^*} has been computed as described in the protocol or, equivalently, as described in the first case of the third round of the simulator. To compute z_j for $j \in [3] \setminus \{i^*\}$, we follow the description of the third round of the simulator, where we simply need to distinguish between the cases $P_j \in M$ and $P_j \in H$. The indistinguishability between this and the previous hybrid follows from the indistinguishability of the outputs generated in the real-world and the ideal-world. We argue this hybrid transition more formally in Lemma A.5.

Hybrid H_9: For every $i \in H$ and $i \neq i^*$, choose z_i uniformly at random and compute z_{i^*} as described in the previous hybrid. This hybrid is identically distributed to the previous one since $\delta_1, \delta_2, \delta_3$ are randomly sampled, i.e. the indistinguishability between this and the previous hybrid follows from security of the one-time pad, with δ as the key, which we prove in Lemma A.6.

Hybrid H_{10}: If $P_3 \in M$ or $P_1 \in H$, this hybrid is skipped. Otherwise, $\overline{\mathsf{ot}_2}$ is generated as $\overline{\mathsf{ot}_2} \leftarrow \mathsf{OT}_2(\mathsf{ot}_1, \mathsf{ot}_2^0, \mathsf{ot}_2^1)$ instead of $\overline{\mathsf{ot}_2} \leftarrow \mathsf{OT}_2(\mathsf{ot}_1, \mathsf{ot}_2^{x_1}, \mathsf{ot}_2^{x_1})$. This hybrid is computationally indistinguishable to the previous hybrid, which can be shown by relying on the sender security of the OT protocol. The proof of the hybrid transition here is similar to the proof of the hybrid transition from H_3 to H_4, therefore, we refer to this proof (Lemma A.4) for further details.

Hybrid H_{11}: If $P_3 \in M$ or $P_2 \in H$, this hybrid is skipped. Otherwise, ot_2^b is generated as $\mathsf{ot}_2^b \leftarrow \mathsf{OT}_2(\mathsf{ot}_1^b, x_{3,0}, x_{3,1})$ instead of $\mathsf{ot}_2^b \leftarrow \mathsf{OT}_2(\mathsf{ot}_1^b, x_{3,x_{2,b}}, x_{3,x_{2,b}})$ for each $b \in \{0,1\}$. This hybrid is computationally indistinguishable to the previous hybrid, which can be shown by relying on the sender security of the OT protocol. The proof of the hybrid transition here is similar to the proof of the hybrid transition from H_3 to H_4, therefore, we refer to this proof (Lemma A.4) for further details.

Hybrid H_{12}: If $P_2 \in M$ or $P_3 \in H$, this hybrid is skipped. Otherwise, ot_2^3 is generated as $\mathsf{ot}_2^3 \leftarrow \mathsf{OT}_2(\mathsf{ot}_1^3, x_{2,0,0}, x_{2,0,1})$ instead of $\mathsf{ot}_2^3 \leftarrow \mathsf{OT}_2(\mathsf{ot}_1^3, x_{2,0,x_3}, x_{2,0,x_3})$. This hybrid is computationally indistinguishable to the previous hybrid, which can be shown by relying on the sender security of the OT protocol. The proof of the hybrid transition here is similar to the proof of the hybrid transition from H_3 to H_4, therefore, we refer to this proof (Lemma A.4) for further details.

Hybrid H_{13}: If $P_2 \in M$ or $P_1 \in H$, this hybrid is skipped. Otherwise, ot_2 is generated as $\mathsf{ot}_2 \leftarrow \mathsf{OT}_2(\mathsf{ot}_1, (x_{2,0}, \mathsf{sk}_0), (x_{2,1}, \mathsf{sk}_1))$ instead of $\mathsf{ot}_2 \leftarrow \mathsf{OT}_2(\mathsf{ot}_1, (x_{2,x_1}, \mathsf{sk}_{x_1}), (x_{2,x_1}, \mathsf{sk}_{x_1}))$. This hybrid is computationally indistinguishable to the hybrid H_3, which can be shown by relying on the weak adaptive sender security of the OT protocol. We prove this transition more formally in Lemma A.4. Since this hybrid is identically distributed to the simulated distribution, the theorem follows.

□

Lemma A.3 (Transition from H_0 to H_1). *Assuming the equivocal receiver security of the OT protocol* OT*, the hybrids* H_0 *and* H_1 *are computationally indistinguishable.*

Proof. We prove this lemma by contradiction, we assume that there exists an adversary $\mathcal{A}$ that manages to distinguish between the hybrids H_0 and H_1 with non-negligible probability. We use this adversary to construct an adversary $\mathcal{B}$ that breaks the equivocal receiver security of the OT protocol OT with non-negligible probability. We assume that $P_1 \in H$, if this is not the case, then this hybrid transition is skipped.

Now, we describe the behavior of the adversary $\mathcal{B}$ interacting with the challenger of the equivocal receiver security of the OT protocol OT and the adversary $\mathcal{A}$. In the first step, the adversary $\mathcal{B}$ samples $(x_1, y_1) \leftarrow \mathcal{D}$, submits x_1 to the underlying challenger and receives as reply the message ot_1 and the key sk corresponding to x_1, i.e. sk_{x_1}. Now, we distinguish between three cases: $P_2 \in H$ and $P_3 \in M$, $P_2 \in M$ and $P_3 \in H$, and, $P_2, P_3 \in M$. In the first and second case, the adversary $\mathcal{B}$ generates the messages of the party P_2 or P_3, respectively, as described in the first round of the protocol in Figure A.3. In the third case, the adversary $\mathcal{B}$ does not do anything on behalf of the parties P_2 and P_3 and simply broadcasts the message ot_1 that it has received from the underlying challenger. For the remaining rounds, the adversary $\mathcal{B}$ behaves as described in the protocol in Figure A.3.

We observe that if the underlying challenger generates the first round of the OT honestly, i.e. $(\mathsf{ot}_1, \mathsf{sk}) \leftarrow \mathsf{OT}_1(1^\lambda, x_1)$, then the hybrid H_0 is emulated and if the underlying challenger generates the first round of the OT using the equivocator, i.e. $(\mathsf{ot}_1, \mathsf{sk}'_0, \mathsf{sk}'_1) \leftarrow \mathsf{Sim}^{Eq}_{\mathsf{OT}}(1^\lambda)$ and $\mathsf{sk} := \mathsf{sk}'_{x_1}$, then the hybrid H_1 is emulated towards $\mathcal{A}$. This directly results in the fact that if $\mathcal{A}$ can distinguish the hybrids H_0 and H_1 with non-negligible probability then $\mathcal{B}$ can also distinguish between an honest and equivocal generation of the first OT message. This concludes the proof of the lemma. □

Lemma A.4 (Transition from H_3 to H_4). *Assuming the sender security of the OT protocol* OT*, the hybrids* H_3 *and* H_4 *are computationally indistinguishable.*

Proof. This hybrid transition only happens under the assumption that the adversary provides consistent transcripts and that $P_2 \in H$ and $P_1 \in M$. Otherwise this hybrid is skipped. Furthermore, we assume that the input x_1 of P_1 is provided to the reduction as an auxiliary input, given a fixed first round of the protocol.

We prove this lemma by contradiction, we assume that there exists an adversary $\mathcal{A}$ that manages to distinguish between the hybrids H_3 and H_4 with non-negligible probability. We use this adversary to construct an adversary $\mathcal{B}$ that breaks the weak adaptive semi-honest sender security of the OT protocol OT with non-negligible probability.

Now, we describe the behavior of the adversary $\mathcal{B}$, which behaves on behalf of P_2, interacting with the challenger of the weak adaptive semi-honest sender security of the OT protocol OT and the adversary $\mathcal{A}$. In the first step, the adversary $\mathcal{B}$ generates ot_1^b using the equivocator for both $b \in \{0,1\}$, i.e., $(\mathsf{ot}_1^b, \mathsf{sk}'_{b,0}, \mathsf{sk}'_{b,1}) \leftarrow \mathsf{Sim}^{Eq}_{\mathsf{OT}}(1^\lambda)$ which it outputs as its first round message. If $P_3 \in H$, it also generates its first round message ot_1^3 using the equivocator, i.e., $(\mathsf{ot}_1^3, \mathsf{sk}''_0, \mathsf{sk}''_1) \leftarrow \mathsf{Sim}^{Eq}_{\mathsf{OT}}(1^\lambda)$. To generate the second round message on behalf of

P_2, $\mathcal{B}$ samples $(x_2, y_2) \leftarrow \mathcal{D}$ and $x_{2,0}, x_{2,1} \leftarrow \{0,1\}$ with the restriction that $x_2 = x_{2,0} + x_{2,1}$. Afterwards, it sends $((x_{2,0}, \mathsf{sk}_0), (x_{2,1}, \mathsf{sk}_1))$ to its underlying challenger and receives as a reply ot_2 which it forwards to the adversary. For the remaining rounds, the adversary $\mathcal{B}$ behaves as described in the protocol in Figure A.3.

We observe that if the underlying challenger generates the second round of the OT using both inputs, i.e. $\mathsf{ot}_2 \leftarrow \mathsf{OT}_2(\mathsf{ot}_1, (x_{2,0}, \mathsf{sk}_0), (x_{2,1}, \mathsf{sk}_1))$, then the hybrid H_3 is emulated and if the underlying challenger generates the second round of the OT using only a single message, i.e. $\mathsf{ot}_2 \leftarrow \mathsf{OT}_2(\mathsf{ot}_1, (x_{2,x_1}, \mathsf{sk}_{x_1}), (x_{2,x_1}, \mathsf{sk}_{x_1}))$, then the hybrid H_4 is emulated towards $\mathcal{A}$. This directly results in the fact that if $\mathcal{A}$ can distinguish the hybrids H_3 and H_4 with non-negligible probability then $\mathcal{B}$ can also break the weak adaptive semi-honest sender security with non-negligible probability. This concludes the proof of the lemma.

□

Lemma A.5 (Transition from H_7 to H_8). *The hybrids H_7 and H_8 are computationally indistinguishable.*

Proof. This hybrid transition only happens under the assumption that the adversary provides consistent transcripts. Otherwise, this hybrid is skipped. Furthermore, we assume that the inputs (x_i, y_i) of P_i for all $i \in M$ is provided to the reduction as an auxiliary input, given a fixed first round of the protocol.

We prove this lemma by contradiction, we assume that there exists an adversary $\mathcal{A}$ that manages to distinguish between the hybrids H_7 and H_8 with non-negligible probability. We use this adversary to construct an adversary $\mathcal{B}$ that distinguishes between the distributions induced by the real-world and the ideal-world.

The adversary $\mathcal{B}$ behaves as described in H_7 until it receives the values $(x_i, y_i)_{i \in M}$ from the adversary $\mathcal{A}$ using its auxiliary input and forwards those values to the underlying challenger. The underlying challenger now samples $\{(x_i, y_i)\}_{i \in H}$ and then either computes z by evaluating $f((x_1, y_1), \ldots, (x_n, y_n))$ or by first generating $\{(x'_i, y'_i)\}_{i \in M} \leftarrow \mathcal{S}_{\mathsf{out}}(\{x_i, y_i\}_{i \in M})$, then computing $z' := f(\{(x_i, y_i), (x'_j, y'_j)\}_{i \in H, j \in M})$ and obtaining $z \leftarrow \mathcal{S}_{\mathsf{out}}(z')$. Then, the value z is output to the adversary $\mathcal{B}$. The adversary then proceeds as described in the description of hybrid H_8 to compute z_{i^*}.

We observe that if the underlying challenger generates the output as described in the real-world, then the hybrid H_7 is emulated and if the underlying challenger generates the output as described in the ideal-world, then the hybrid H_8 is emulated towards $\mathcal{A}$. This directly results in the fact that if $\mathcal{A}$ can distinguish the hybrids H_7 and H_8 with non-negligible probability then $\mathcal{B}$ can also distinguish between the real- and ideal-world. This concludes the proof of the lemma. □

Lemma A.6 (Transition from H_8 to H_9). *The hybrids H_8 and H_9 are perfectly indistinguishable.*

Proof. This hybrid transition only happens under the assumption that the adversary provides consistent transcripts. Otherwise, this hybrid is skipped. Furthermore, we assume that the input x_1 of P_1 is provided to the reduction as an auxiliary input, given a fixed first round of the protocol.

To prove this lemma, we distinguish between the following cases:

$P_1 \in H$: In this case, the values z_i for all $i \in [3] \setminus (\{1\} \cup H)$ are computed as described in round 3 of the simulator (Figure A.4) for the malicious P_i. To compute the value for the parties P_i for $i \in H \setminus \{1\}$ we interact with a one-time pad challenger and submit the query $(x_{2,0,0}+y_2, 0)$ if $P_2 \in H$ or $(x_{2,0,x_3}+x_3, y_3, 0)$ if $P_3 \in H$. The reply received from this query is either the value z_2 or z_3 respectively. Afterwards, the value z_1 is computed as $z_1 := z \bigoplus_{i \in [3] \setminus \{1\}} z_i$ and broadcast by P_1. From the security of the one-time pad, it follows that the value z_i for $i \in H \setminus \{1\}$ is independent of the input x_i and y_i.

$P_1 \notin H$ and $P_2 \in H$: In this case, the values z_i for all $i \in [3] \setminus (\{2\} \cup H)$ are computed as described in round 3 of the simulator (Figure A.4) for the malicious P_i. To compute the value for the parties P_i for $i \in H \setminus \{i\}$ we interact with a one-time challenger and submit the query $(x_{2,0,x_3} + x_3, y_3, 0)$ if $P_3 \in H$. The reply received from this query is the value z_3. Afterwards, the value z_2 is computed as $z_2 := z \bigoplus_{i \in [3] \setminus \{2\}} z_i$ and broadcast by P_2. From the security of the one-time pad, it follows that the value z_i for $i \in H \setminus \{2\}$ is independent of the input x_i and y_i.

$P_1, P_2 \notin H$ and $P_3 \in H$: In this case, no more changes are needed compared to the previous hybrid and the lemma follows directly.

□

A.3 From 3MULTPlus to General MPC Functionality

In this section we recap the transformation of [BGI+18, GIS18, ABG+20] and argue that extends to Definition A.1. Let us start by observing that the Lemma 6.1 of [ABG+20] could also be proven in the case of Definition A.1.

Following [ABG+20] we consider polynomials over mn input variables, where each party contributes m inputs. We assume that the polynomial is over $\mathbb{Z}_2$ and it is the sum of monomials of degree exactly 3. Let $\mathcal{D}_{\mathsf{inp}}$ be the distribution of the parties' input. Let $\mathcal{D}$ be the distribution that draws the constant term 0 and then samples randomly from $\{0,1\}$. Let $\mathcal{D}'$ be the distribution that samples two elements one from $\mathcal{D}_{\mathsf{inp}}$ and one from the uniform distribution over $\{0,1\}$. Let $\mathcal{D}_{\mathsf{3deg}}$ be the distribution that draws from $\mathcal{D}$ and $\mathcal{D}'$.

Lemma A.7. *Let $g : \{0,1\}^{mn} \rightarrow \{0,1\}$ be a degree-3 functionality over $\mathbb{Z}_2$. That is, $g((x_1,\dots,x_m),\dots,(x_{mn-m+1},\dots,x_{mn})) = \sum a_{i_1\dots i_3} x_{i_1} x_{i_2} x_{i_3}$. There exists a protocol Π_{3deg} for computing g, satisfying Definition A.1, where the protocol Π_{3deg} makes black-box use of the protocol $\Pi_{\mathsf{3MULTPlus}}$ (described in Figure A.3) for distribution $\mathcal{D}_{\mathsf{3deg}}$ for* 3MULTPlus *functionality satisfying Definition A.1 for distributions $\mathcal{D}$ and $\mathcal{D}'$.*

Proof. The proof is adapted from [ABG+20]. For every 3-sized subset $\{i_1, i_2, i_3\} \in [mn]$, the parties $P_{\lfloor i_j/m \rfloor+1}$, where $j \in [3]$ do the following:

- Party $P_{\lfloor i_j/m \rfloor+1}$ chooses a random bit $r^{i_j}_{i_1,i_2,i_3}$.
- If $a_{i_1\dots i_3} = 0$ party $P_{\lfloor i_j/m \rfloor+1}$ on input $(0, r^{i_j}_{i_1,i_2,i_3})$ execute $\Pi_{\mathsf{3MULTPlus}}$, for $j \in [3]$.
- Else, party $P_{\lfloor i_j/m \rfloor+1}$ on input $(x_{i_j}, r^{i_j}_{i_1,i_2,i_3})$ execute $\Pi_{\mathsf{3MULTPlus}}$, for $j \in [3]$.

The parties then sum up their chosen random bits, namely $\sum_{j=1}^{m} r^{i_j}_{i_1,i_2,i_3}$, and send them to every other party. To obtain the output, the parties sum up the outputs from the executions of $\Pi_{\mathsf{3MULTPlus}}$ with the sum of all the random bits obtained from every party.

The correctness follows from [ABG+20]. Regarding security, let $M \subseteq [n]$ be the set of corrupted parties. Let Sim be the simulator described in Figure A.4 for $\Pi_{\mathsf{3MULTPlus}}$. The simulator Sim' for Π_{3deg} follows the same steps (for all the executions of $\Pi_{\mathsf{3MULTPlus}}$) of Sim for the first and second rounds. At the end of the second round, Sim' obtains adversarial inputs $(x_{(i-1)m+1},\dots,x_{im})$ and random tapes, for $i \in M$ and distinguishes two cases:

(a) **$\mathcal{A}$ behaved inconsistently in the first two rounds** Sim' follows the strategies of Sim described in Round 3 (a) (Figure A.4) for all executions of $\Pi_{\mathsf{3MULTPlus}}$.

(b) **Otherwise** Sim' queries the ideal functionality obtaining the output g. For every 3-sized subset $\{i_1, i_2, i_3\}$ such that for each $j \in [3]$, $\lfloor i_j/m \rfloor + 1 \in M$ and $a_{i_1\dots i_3} = 1$, the simulator adds the corresponding adversarial inputs to g. Let g' be the resulting value. Let $i^* \notin M$ be an arbitrary element. For every $i \neq i^*$ and $i \notin M$ Sim sends a random bit as the sum of P_i's random bits. For every 3-sized subset $\{i_1, i_2, i_3\}$ such that for exists $j \in [3]$, $\lfloor i_j/m \rfloor + 1 \notin M$ Sim' follows the steps of Sim w.r.t. a random output. For i^*, the simulator outputs g' plus the sum of all the random bits chosen as the sum of its random bits.

The set of hybrids to consider for the transition from real-world execution to the experiment where Sim' is executed is similar to the one described for Theorem A.2. In more detail, the hybrid H_0 is described as the hybrid H_0 of Theorem A.2. Then, let k the number of executions of $\Pi_{\mathsf{3MULTPlus}}$, then we consider a set of k hybrids where H^j_i, described as the hybrid H_i of Theorem A.2 but w.r.t. the j-th executions of $\Pi_{\mathsf{3MULTPlus}}$ (where the output of $\Pi_{\mathsf{3MULTPlus}}$ is a random bit), with $j \in [k]$ and $i \in [4]$. The final hybrid sends a random bit as the sum of P_i's random bits (for all $i \notin M$) and adjusts the output of the honest parties as Sim' does.

The indistinguishability between the hybrids is argued similarly to Theorem A.2. □

We are now ready to argue that the transformation of [BGI+18, GIS18, ABG+20] extends to Definition A.1. Let $\mathcal{D}_{\mathsf{inp}}$ be the distribution of the parties' input. Let $\mathcal{D}$ be the distribution that samples two elements from the uniform distribution over $\{0,1\}$ one from $\mathcal{D}_{\mathsf{inp}}$ and the constant terms $0, 1, 0$. Let $\mathcal{D}_f$ be the distribution that draws from $\mathcal{D}$ and $\mathcal{D}_{\mathsf{3deg}}$.

Theorem A.8 (**[BGI⁺18, GIS18, ABG⁺20]**). *Let f be an n-party functionality. There exists a protocol for securely computing f satisfying Definition 6.1 w.r.t. distributions $\mathcal{D}_f$, where Π_f makes black-box use of the PRG and the protocols: (a) the protocol $\Pi_{\mathsf{3MULTPlus}}$ (described in Figure A.3) for* 3MULTPlus *functionality satisfying Definition A.1 w.r.t. distributions $\mathcal{D}$ (b) the protocol Π_{3deg} (Lemma A.8) for degree-3 functionality satisfying Definition A.1 w.r.t. distributions $\mathcal{D}_{\mathsf{3deg}}$.*

Proof. The proof is adapted from [ABG⁺20] (which is adapted from [GIS18]). Assume that f is represented by a circuit comprised entirely of NAND gates. We recall the semantics of a BMR garbled gate. The BMR garbling for a NAND gate g that takes wires a and b as input and the output wire is c is a set of values $\{\tilde{G}^i_{r_1,r_2}\}_{r_1,r_2\in\{0,1\},i\in[n]}$ where:

$$\tilde{G}^i_{r_1,r_2} = \left(\bigoplus_{i=1}^{n} F_{k^i_{a,r_1}}(g,i,r_1,r_2) \oplus F_{k^i_{b,r_1}}(g,i,r_1,r_2)\right) \oplus k^i_{c,0} \oplus (X_{r_1,r_2} \wedge (k^i_{c,1} \oplus k^i_{c,0}))$$

where $X_{r_1,r_2} = ((\bigoplus_{i=1}^n \lambda_{i,a} \oplus r_1) \cdot (\bigoplus_{i=1}^n \lambda_{i,b} \oplus r_2) \oplus 1) \oplus (\bigoplus_{i=1}^n \lambda_{i,c})$. Here, F is a PRF, $k^i_{x,r}$ where $x \in \{a,b,c\}$ and $r \in \{0,1\}$ is a PRF key, $\lambda_{i,x}$ for $x \in \{a,b,c\}$ are bits. The PRF keys $k^i_{x,r}$ and the bits $\lambda_{i,x}$ are chosen by each party before the first round of the protocol.

Note that each output bit of $\{\tilde{G}^i_{r_1,r_2}\}_{r_1,r_2\in\{0,1\},i\in[n]}$ is a degree-3 functionality.

The protocol Π_f has the following description:

- For every wire w, which is the input wire of a party P_i the other parties P_j will set $\lambda_{j,w} = 0$. The party P_i will compute $\alpha_w = \lambda_{j,w} \oplus x_w$, where x_w is the w-th bit of P_i's input x_i.
- For every α_w, the party P_j, the party P who owns w and a dummy party P' execute $\Pi_{\mathsf{3MULTPlus}}$ on input $((k^i_{w,0} \oplus k^i_{w,1}, k^i_{w,0})(\alpha_w, 0)(1,0))$. All the parties receive k^i_{w,α_w}.
- For every NAND gate g in f, the parties uses Π_{3deg} to compute $\{\tilde{G}^i_{r_1,r_2}\}_{r_1,r_2\in\{0,1\},i\in[n]}$.
- Evaluate the BMR garbled circuit to obtain the output.

As noticed by [ABG⁺20] in $\{\tilde{G}^i_{r_1,r_2}\}_{r_1,r_2\in\{0,1\},i\in[n]}$ together with $\{k^i_{w,\alpha_w}, \alpha_w\}_{w\in\mathsf{inp}}$, constitutes a computationally private randomized encoding [BMR90]. Let $\mathsf{Sim}_{\mathsf{RE}}$ be the simulator of the randomized encoding. Let $\mathsf{Sim}_{\mathsf{3MULTPlus}}$, $\mathsf{Sim}_{\mathsf{3deg}}$ be the simulators, respectively, for $\Pi_{\mathsf{3MULTPlus}}$ and Π_{3deg} (Lemma A.8). Let $M \subseteq [n]$ be the set of corrupted parties.

The simulator Sim for Π_f follows the same steps (for all the executions of $\Pi_{\mathsf{3MULTPlus}}$ and of Π_{3deg}) of $\mathsf{Sim}_{\mathsf{3MULTPlus}}$ and of $\mathsf{Sim}_{\mathsf{3deg}}$ for the first and second rounds. At the end of the second round, Sim obtains adversarial inputs $\{x_i\}_{i\in M}$ and random tapes and distinguishes two cases:

$\mathcal{A}$ behaved inconsistently in the first two rounds Sim follows the strategies of $\mathsf{Sim}_{\mathsf{3MULTPlus}}$ described in Round 3 (a) (Figure A.4) for all executions of $\Pi_{\mathsf{3MULTPlus}}$, and the strategies of $\mathsf{Sim}_{\mathsf{3deg}}$ described in point (a) of Lemma A.8 for all executions of Π_{3deg}.

Otherwise Sim queries the ideal functionality obtaining the output out. For every NAND gate g in f, Sim follows the strategies of $\mathsf{Sim}_{\mathsf{RE}}$ (executed on input out and $\{x_i\}_{i\in M}$) to compute the output of $\Pi_{\mathsf{3MULTPlus}}$ and Π_{3deg}, then she follows the strategies $\mathsf{Sim}_{\mathsf{3MULTPlus}}$ and of $\mathsf{Sim}_{\mathsf{3deg}}$ to set those outputs as the output of the protocols $\Pi_{\mathsf{3MULTPlus}}$ and Π_{3deg}.

The indistinguishability between the hybrids follows from similar arguments to the one exposed for Lemma A.8 and from the fact that $\{\tilde{G}^i_{r_1,r_2}\}_{r_1,r_2\in\{0,1\},i\in[n]}$ together with $\{k^i_{w,\alpha_w}, \alpha_w\}_{w\in\mathsf{inp}}$, constitutes a computationally private randomized encoding.

□