Round-Optimal Multi-Party Computation with Identifiable Abort

Michele Ciampi[1], Divya Ravi[2] ⋆,
Luisa Siniscalchi[2,3], and Hendrik Waldner[4,5]

[1] The University of Edinburgh, Edinburgh, UK
michele.ciampi@ed.ac.uk
[2] Aarhus University, Aarhus, Denmark
{divya,lsiniscalchi}@cs.au.dk
[3] Concordium Blockchain Research Center, Aarhus, Denmark
[4] Max Planck Institute for Security and Privacy, Bochum, Germany
hendrik.waldner@mpi-sp.org
[5] University of Maryland, College Park, US

Abstract. Secure multi-party computation (MPC) protocols that are resilient to a dishonest majority allow the adversary to get the output of the computation while, at the same time, forcing the honest parties to abort. Aumann and Lindell introduced the enhanced notion of *security with identifiable abort*, which still allows the adversary to trigger an abort but, at the same time, it enables the honest parties to agree on the identity of the party that led to the abort. More recently, in Eurocrypt 2016, Garg et al. showed that, assuming access to a simultaneous message exchange channel for all the parties, at least four rounds of communication are required to securely realize non-trivial functionalities in the plain model.

Following Garg et al., a sequence of works has matched this lower bound, but none of them achieved security with identifiable abort. In this work, we close this gap and show that four rounds of communication are also sufficient to securely realize any functionality with identifiable abort using standard and generic polynomial-time assumptions. To achieve this result we introduce the new notion of *bounded-rewind secure MPC* that guarantees security even against an adversary that performs a mild form of reset attacks. We show how to instantiate this primitive starting from *any* MPC protocol and by assuming trapdoor-permutations.

The notion of bounded-rewind secure MPC allows for easier parallel composition of MPC protocols with other (interactive) cryptographic primitives. Therefore, we believe that this primitive can be useful in other contexts in which it is crucial to combine multiple primitives with MPC protocols while keeping the round complexity of the final protocol low.

⋆ Funded by the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC).

1 Introduction

Secure multi-party computation (MPC) [25,46] 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 [2,6,7,11,12,22,25,26,33–35, 37,39,41,45].

One interesting line of research concerns proving the security of MPC protocols in the case of a dishonest majority. In this model, unfortunately, it is in general impossible to obtain *guaranteed output delivery* or even *fairness* [13], which are particularly useful properties. The former guarantees that the honest parties always receive the output of the computation and the latter guarantees that either all the parties receive the output or none does (not even the corrupted parties). Due to the impossibility of Cleve et al. [13], most of the MPC protocols proven secure in the dishonest majority setting only satisfy the notion of *unanimous abort*. This notion guarantees that either all the honest parties receive the output, or none of them does. Another recent line of works has established that *four rounds* are both necessary [22] and sufficient [2,4,7,10,29] for MPC with unanimous abort (with respect to black-box simulation) while relying on broadcast.[6] However, none of these works study the notion of *MPC with identifiable abort*.

The notion of MPC with identifiable abort, which was first considered by Aumann and Lindell [3], ensures that either the honest parties receive the output of the computation or they unanimously identify the (corrupted) party that led to the abort. Subsequently, Ishai, Ostrovsky, and Zikas [32] showed how to achieve this notion in the information theoretic and computational setting, and propose a construction (in the computational setting) that does not rely on any setup assumptions. The work of Ishai et al. led to a sequence of works that proposed improved protocols realizing security with identifiable abort [5, 8,9,16,43]. All of these works either require a trusted setup (e.g., correlated randomness) or require more than a constant number of rounds.

Moreover, the new recent lower bounds on MPC with unanimous abort and the new results on MPC with identifiable abort leave open the following question:

> *What is the best-possible round complexity for securely evaluating any function with identifiable abort (with black-box simulation) in the plain model when the majority of the parties are corrupted and broadcast channels are assumed?*

In this work, we answer the above question[7] and match the lower bound proven in [22] by presenting a four-round protocol with identifiable abort rely-

[6] In each round all the parties can send a message. That is, the channel allows for a simultaneous exchange of messages in each round. Unless otherwise specified we implicitly refer to this model of communication when referring to broadcast.

[7] All our results are with respect to black-box simulation. Hereafter we assume that this is implicitly stated in our claims.

ing only on standard polynomial-time cryptographic assumptions (i.e., one-way trapdoor permutations).[8] To the best of our knowledge, prior to our work, no protocol achieved security with identifiable abort in the plain model in four rounds of communication.

To achieve this result, we define and construct a four-round *bounded-rewind secure MPC* protocol as an intermediate step. A bounded-rewind secure MPC protocol enjoys the same security as a standard MPC protocol (with unanimous abort), and is additionally resilient to *rewind attacks*. More precisely, the protocol remains secure even if the adversary is allowed to receive a (bounded) number of third rounds in response to multiple (adversarially chosen) second-round messages. The reason why we define (and construct a protocol that satisfies) this new security notion is to obtain an MPC protocol that can be easily composed in parallel with other interactive cryptographic primitives. We see this as a result of independent interest and we believe that this notion is also useful in other contexts where MPC protocols need to be combined with other interactive primitives (e.g., key distribution protocols, proof-of-knowledge or even other MPC protocols). This notion becomes instrumental in our construction, since we will execute an MPC protocol and a zero-knowledge-like protocol in parallel. We give more details on the notion and the constructions later in this section.

1.1 Our results

As previously mentioned, the stepping stone of our construction is a four-round bounded-rewind secure MPC protocol. We realize this notion using a compiler that turns, in a round-preserving manner, *any* four-round MPC protocol into a bounded-rewind secure MPC protocol. Our compiler relies on a bounded-rewind secure oblivious transfer (OT) protocol (similar to the one proposed by Choudhuri et al. [10]), on Yao's Garbled Circuits (GC) and public-key encryption. Unfortunately, we cannot directly use the OT protocol of Choudhuri et al. [10], since we need the OT to be simulation-based secure against malicious receivers.[9] Hence, we also need to prove that such a bounded-rewind secure OT protocol is indeed simulatable. The bounded-rewind secure OT protocol and the public-key encryption scheme can be instantiated from trapdoor permutations (TDPs), and GCs can be based on one-way functions. Given the above, we can claim the following:

Theorem 1 (informal). *Assuming TDPs and the existence of a* 4*-round MPC protocol that realizes the function f with unanimous abort over broadcast channels against a dishonest majority, then there exists a bounded-rewind secure* 4*-round MPC protocol that realizes the same function (relying on the same communication channel) with unanimous abort against a dishonest majority.*

[8] Some of the tools used in our constructions require the trapdoor permutations to be certifiable. Any time that we refer to trapdoor permutations we implicitly assume that they are certifiable. Note that such trapdoor permutations can be instantiated using RSA with suitable parameters [21].

[9] We require an additional property on the OT, which we elaborate further in the next section.

To finally obtain our round-optimal MPC protocol that is secure with identifiable abort, we compose the bounded-rewind secure MPC protocol, in parallel, with a combination of two-round witness-indistinguishable proofs, signature schemes and three-round non-malleable commitment schemes. We provide more detail regarding this in the next section. To obtain our final construction we require the bounded-rewind secure MPC protocol to be perfectly correct. Finally, observing that all the additional tools we need can be based on TDPs, we can claim the following.

Theorem 2 (informal). *Assuming TDPs and the existence of a perfectly correct* 4*-round bounded-rewind secure MPC protocol that realizes the function* f *with* unanimous abort *over broadcast channels against a dishonest majority, then there exists a* 4*-round MPC protocol that realizes the same function (relying on the same communication channel) with* identifiable abort *against a dishonest majority.*

To state our main theorem, we argue that the four-round MPC protocol proposed in [10] is perfectly correct and that the final bounded-rewind MPC protocol we obtain from Theorem 1 preserves the perfect correctness of the input protocol. Given that the protocol of Choudhuri et al. [10] is based on OT, which in turn can be based on TDPs, we can state the following.

Corollary (informal). *Assuming TDPs then there exists a* 4*-round MPC protocol that realizes any function* f *with identifiable abort over broadcast channels against a dishonest majority.*

1.2 Technical Overview

The Challenge of Obtaining MPC with Identifiable abort. Among many other interesting results, in [32] the authors propose a protocol that realizes any function with identifiable abort in the plain model. In more detail, Ishai et al. propose a generic approach to turn any MPC protocol with identifiable abort, that relies on correlated randomness as a setup assumption, into a secure MPC protocol with identifiable abort in the plain model. Their compiler is quite straightforward: the parties use an MPC protocol Π^{CR} to generate correlated randomness, which is then used to run the previously mentioned protocol of [32], which we denote as Π^{IOZ}. In the case that some parties abort during the execution of Π^{IOZ} the property of identifiable abort is trivially maintained since Π^{IOZ} is proven secure under the assumption that correlated randomness exists which, in turn, has been generated using by the protocol Π^{CR}. On the other hand, if an abort occurs during the execution of Π^{CR} then all the parties could simply disclose the randomness used to run Π^{CR} and check which party did not follow the protocol description. Note that the randomness of the parties can be disclosed at this stage of the protocol since Π^{CR} does not require the parties' input to be executed and therefore privacy is still guaranteed. However, such an approach crucially needs the protocol Π^{CR} to be secure against *adaptive* corruptions.[10] Indeed, without this property it is not clear how to prove the security of the protocol. This is due to the fact that, during the simulation of Π^{CR}, it might be necessary to output the random coins used by the honest parties, which are controlled by the simulator. Revealing the random coins of the simulator might make the simulated execution trivially distinguishable from the real one. Unfortunately, it is not clear how to use such an approach to obtain a constant round protocol since it has been shown in [23] that it is impossible to achieve security with adaptive corruptions in a constant number of rounds in the plain model.

Another approach that one could follow is to start from a protocol Π that is proven secure in the plain model and attach a public coin zero-knowledge (ZK) proof to each round of Π. That is, each party computes one message of Π and then runs a ZK proof to show that the computed message has been generated accordingly to Π. We first note that such an approach does not immediately work if Π is not perfectly correct. Indeed, if Π is not perfectly correct then there might exist randomness that, if used to compute a message of Π, would make the receiver of this message abort. However, the adversary would be able to complete the proof since it has followed the protocol. Instead of using a perfectly correct protocol Π one could add a coin-tossing protocol, but this would create additional issues since the coin-tossing protocol needs to be secure with identifiable abort.

Another issue with the above approach is that, even in the case where Π is perfectly correct, we cannot just use a standard public-coin ZK proof, given that the adversary might maul the ZK proof received from an honest party. To account for this, using a public-coin non-malleable ZK in combination with a perfectly correct Π seems to be a reasonable direction. But, also in this case, if we want a constant round protocol we need to require the public-coin ZK to be executable in a constant number of rounds and, as shown in [24], only trivial languages admit constant round public-coin black-box ZK protocols (with negligible soundness error). Therefore, if we want to use such an approach, we need to relax the public-coin requirement, and, indeed, *public verifiability* suffices. We say that a ZK protocol is publicly verifiable if, by looking at the messages of the protocol exchanged between a prover and a verifier, it is possible to infer whether the honest verifier would accept the proof without knowing its random coins. Moreover, it must be possible to detect whether the verifier is sending valid messages (i.e., messages that would not make the honest prover abort) without knowing the randomness of the prover and by just looking at the transcript. This property is particularly important for our purposes since it allows a party P, that is not involved in the execution of the non-malleable ZK protocol between two parties (in which one is acting as a prover and the other as the verifier), to detect which party caused the abort (if any). If the prover is malicious and the verifier rejects, then P notices this and it can tag the party acting as

[10] In this model, the identities of the corrupted parties are not fixed at the beginning of the experiment and the adversary can decide which party to corrupt during the execution of the protocol.

the prover as being corrupted. If instead the verifier sends a message that the prover would reject then, also due to the public verifiability, P can tag the party acting as the verifier as corrupted.

Assuming that there are constant round ZK protocols with all these properties (hereafter we refer to them as *special ZK* protocols) and that Π is a perfectly correct constant round protocol, we can finally construct an MPC protocol with identifiable abort (in a constant number of rounds) in the plain model.

However, in this work, we want to study the optimal round complexity for MPC with identifiable abort. In particular, we want to prove that four rounds are sufficient to securely realize any function with identifiable abort. We start by observing that it is not needed to run a ZK proof after each round of the protocol. Indeed, we could just let the parties run the protocol Π and only at the end, in the case a party aborts, each party generates its zero-knowledge proof (proving that all the messages of Π that it has sent over the broadcast channel have been computed correctly). If the ZK protocol is four-round (which is the best we can hope for) and Π needs four rounds as well (which, again, is the best we can hope for) then we have obtained an 8-round MPC protocol with identifiable abort.

The next natural step, to reduce the round complexity of the above protocol, is to parallelize the messages of Π and the messages of the special ZK protocol. This natural approach fails for two reasons. First, the special ZK protocol now needs to be delayed-input. That is, the statement the parties prove in the case someone aborts is not defined until the fourth round and, second, there is no reason to expect that Π and a zero-knowledge protocol would compose in parallel. Even if there are four-round special ZK protocols that enjoy the property of delayed-input, like the one in [11], at the same time it is unclear how to prove that this protocol composes with Π due to well-known *rewinding issues*. Indeed, one approach to prove the security of this candidate MPC protocol would be to consider a first hybrid experiment in which we run the simulator of the special ZK proof. This step is a straightforward reduction and does not seem to cause issues. Note that, in this intermediate hybrid experiment, the simulator of the ZK protocol is rewinding the adversary and, in particular, we can assume that the simulator rewinds (at least) from the third to the second round. We now proceed to the next hybrid where we run the simulator of Π. Proving the indistinguishability between the two hybrid experiments is problematic. The reason is that the rewinds made by the ZK simulator could make the adversary ask for multiple second rounds with respect to Π. However, the reduction can only receive one set of second round messages from the challenger and it is unclear whether the reduction can fake these messages of Π during the rewinds.

In this work, we solve this issue by constructing an MPC protocol (with unanimous abort) that is *bounded-rewind secure*. That is, such an MPC protocol remains secure even if an adversary asks to receive multiple third-round messages as a reply to multiple (adversarially generated) second round messages. Equipped with this tool we can make the reduction work and complete the proof. We note that this approach works only under the assumption that the ZK simulator rewinds from the third to the second round, which we will argue to be sufficient for our construction. In more detail, in this work we simply combine the bounded-rewind secure MPC protocol with witness-indistinguishable proofs and non-malleable commitments. These tools guarantee a mild form of non-malleability (which is sufficient for our purposes) that is achieved by requiring rewinds only from the third to the second round in the security proof. The way we achieve non-malleability has become quite standard recently and has been used in [4,10,11,22]. For this reason, we do not give more details on these aspects and refer the interested reader to the technical section. Instead, we dedicate the last part of this section to explain how we construct our bounded-rewind secure MPC protocol.

Bounded-Rewind Secure MPC. Our compiler turns a four-round MPC protocol Π, that only relies on a broadcast channel, into a four-round bounded-rewind-secure MPC protocol Π_{rmpc} (that, again, works over a broadcast channel). We start by discussing how the protocol works for the two-party case (with parties P_1 and P_2), and then discuss how to extend our approach to the multi-party case. For reference, in Fig. 1 we provide a pictorial description of the protocol Π_{rmpc}.

As already mentioned, a protocol is bounded-rewind secure if it retains its security even in the case that the adversary queries the honest party on multiple second rounds, and receives an honestly generated third round for each of these queries. It is easy to imagine that most of the existing four-round MPC protocols have no resiliency against such types of attacks. Indeed, usually the simulation strategy adopted to prove the security of these protocols is to rewind from the third to the second round and extract the input of the corrupted parties. Regardless of that, we aim to provide a compiler that works on any four-round MPC protocol without making any additional assumptions on the input protocol.

To prevent the adversary from gaining an advantage, using its rewinds, we adopt a strategy to hide the third round message of Π and only reveal it in the fourth round. To do that we follow an approach similar to [1, 14, 18], by embedding the next-message function of Π inside a garbled circuit (GC). More precisely, each party (e.g., P_1) upon receiving the second round message of Π creates a GC that contains all the messages of Π generated so far, its input and randomness. Note that the GC embeds almost all the information needed to compute the fourth round of Π. The only thing that is missing is the third round of the other party (P_2 in our example). The GC, on input the third round message of P_2 for Π, runs the next-message function and returns the fourth round of Π for P_1.

As one might expect, to securely evaluate the GC, P_1 and P_2 need to run an OT protocol in which, in this example, P_2 acts as the receiver and P_1 acts as the sender. The input of P_1 (which acts as a sender) to the OT protocol are the labels of the GC we have just described, while the input of P_2 (which acts as the receiver) is its third round of the protocol Π. In other words, in the third round of the protocol we have just described each party does not send the third

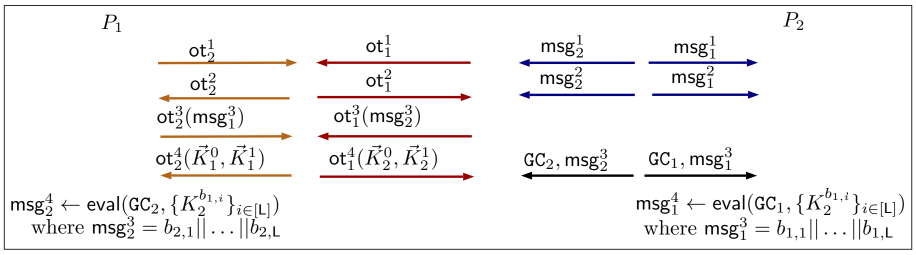

P_1
P_2
ot_2^1
ot_1^1
msg_2^1
msg_1^1
ot_2^2
ot_1^2
msg_2^2
msg_1^2
$\mathsf{ot}_2^3(\mathsf{msg}_1^3)$
$\mathsf{ot}_1^3(\mathsf{msg}_2^3)$
$\mathsf{ot}_2^4(\vec{K}_1^0, \vec{K}_1^1)$
$\mathsf{ot}_1^4(\vec{K}_2^0, \vec{K}_2^1)$
$\mathtt{GC}_2, \mathsf{msg}_2^3$
$\mathtt{GC}_1, \mathsf{msg}_1^3$
$\mathsf{msg}_2^4 \leftarrow \mathsf{eval}(\mathtt{GC}_2, \{K_2^{b_{1,i}}\}_{i \in [\mathsf{L}]})$
where $\mathsf{msg}_2^3 = b_{2,1}|| \dots ||b_{2,\mathsf{L}}$
$\mathsf{msg}_1^4 \leftarrow \mathsf{eval}(\mathtt{GC}_1, \{K_2^{b_{1,i}}\}_{i \in [\mathsf{L}]})$
where $\mathsf{msg}_1^3 = b_{1,1}|| \dots ||b_{1,\mathsf{L}}$

Fig. 1: Our rewindable-secure protocol for the two-party case. msg_i^j represents the j-th message of the MPC protocol Π computed by the party P_i. ot_i^j represents the j-th message of the bounded-rewind secure OT protocol in which P_i acts as the sender and P_{3-i} acts as the receiver with $i \in [2]$. The pairs of inputs $(\vec{K}_i^0, \vec{K}_i^1)$ that the party P_i uses as input when acting as the sender of the OT represent the labels of the wire for the garbled circuit that computes the next-message function of Π. This garbled circuit $\mathtt{GC}_i$ has hardwired-in the input of P_i, its randomness and all the messages of P_i generated up to the second round. The algorithm eval is the GC evaluation algorithm, that on input the encoding of the GC and a set of labels (one per each wire) returns the output of the GC (the last message of Π in this case). Note that the parties in the last round also send the third message of Π. This is because to compute the output of Π the parties might need all the messages generated from Π.

round message of Π over the channel, but it sends the OT receiver message which encodes the third round of Π.

The above approach, however, has an issue. To prove its security we need to use an OT protocol that is simulation-based secure against malicious receivers. This is required because in the simulation we need to use the OT simulator to extract the third round of Π (and forward it to the simulator of Π). Existing OTs that achieve this property require at least four rounds, and this means that our construction is not secure unless the OT protocol is resilient against rewinding attacks. Interestingly, in [10] the authors propose a four-round OT protocol that is secure even in the presence of an adversary that does a bounded number of rewinds. One drawback of the protocol proposed in [10] is that it is not proved to be simulation-based secure against malicious receivers. This should not come as a surprise since it seems to be contradictory to have a primitive that allows extraction through rewinds (since we are in the plain model), but at the same time is secure against adversaries that make rewinds. Fortunately, we can prove that the protocol of [10] is also simulation-based secure against malicious receivers. This proof requires a non-trivial simulator and analysis to argue that the simulated transcript remains indistinguishable from a real one. Our OT simulator and proof crucially rely on the elegant analysis of the simulator for the zero-knowledge protocol proposed by Hazay et al. [30]. We refer the reader to Section 6 for more details.

The multi-party case. A natural extension of the above 2-party approach to the n-party case would be for each pair of parties to engage in an OT instance as a receiver and sender respectively, to retrieve the labels which are needed to query each other GC and to obtain the fourth round of Π. More precisely, each party now prepares a garbled circuit as before with the difference that its GC now accepts $n-1$ inputs ($n-1$ third round messages of the remaining $n-1$ parties).

This approach does not immediately work since, for example, the party P_i would be able to get only the labels for the wires of the garbled circuit of P_j that encodes its own third round. However, to query the garbled circuit of P_j, the party P_i needs at least one label per wire. To allow P_i to get those labels, we use an approach similar to the one proposed in [10], in which, in the fourth round, all the parties broadcast the randomness and the input used when acting as the OT receiver. In this way P_i can finally query P_j's garbled circuit since it has the labels that correspond to the third round messages of all the parties.

However, since we need to rely on the rewindable-security of the OT protocol, we cannot simply let the parties disclose their randomness contrary to what happens in [10]. For this reason we propose a simple modification of the rewind-secure OT that retains an adequate level of security even in the case that part of the randomness used in the computation is disclosed.

Even if the above approach looks promising, it is vulnerable to the following attack by a potentially corrupted $P^\star$: $P^\star$ could use different third-round messages in each of the n OT instances when acting as the OT receiver. This behaviour is problematic since it allows the adversary to recover the fourth-round messages of the honest parties (via the evaluation of their respective garbled circuits) computed with respect to different third-round messages, which could compromise the security of the underlying MPC protocol Π. Indeed, in the case that Π is normally executed over a broadcast channel, honest parties compute their fourth-round message with respect to the same third-round messages.

To solve this problem, we break this one-to-one dependency such that the labels of the garbled circuit (GC) are secret shared among the OT executions in such a way that it is guaranteed that the labels of a party's GC can only be reconstructed if and only if each party has used the same input across the OT executions where it was acting as the receiver. For more details on how this secret sharing works we refer the reader to Section 4.

1.3 Related Work

As already mentioned, the notion of security with identifiable abort was first considered by Aumann and Lindell [3]. This notion was subsequently studied in [15,31,32].

Ishai et al. [32] show that in the correlated randomness model MPC with identifiable abort can be realized information-theoretically. In the information-theoretic setting Ishai et al. require all the n parties to be in possession of some shared randomness and leave open the question of whether information-theoretic ID-MPC can be realized assuming oracles that return correlated randomness shares to less than n parties. This question has been answered in the affirmative in recent works [8, 9, 43].

The idea of using the SPDZ protocol [17, 19] to realize ID-MPC has been used in a few follow-up works. In the work of Spini and Fehr [44] the authors aim to adapt the SDPZ protocol to allow for identifiable abort without increasing the complexity of the protocol too much. Their protocol achieves a communication and computation complexity that polynomially depends on the number of the participating parties. In another work by Cunningham et al. [16], the authors extend the results of Spini and Fehr and obtain a protocol that requires n messages instead of $O(n)$ messages [16, Table 1], where n is the number of parties. Furthermore, their protocol also realizes the notion of *completely* identifiable abort, which is introduced in the same work. The notion of completely identifiable abort extends the existing notion of identifiable abort by not only guaranteeing that a single cheating party is identified but that all the cheating parties are identified.

In the work of Scholl et al. [42] the authors present a compiler that takes any passively secure preprocessing protocol and turns it into one with covert security and identifiable abort. A protocol that fulfills these conditions is, again, the SPDZ protocol [17, 19]. In Baum et al. [5] the authors present a constant round ID-MPC protocol with concrete efficiency. Their protocol only makes black-box use of OT and a circular 2-correlation robust hash function. The security of their protocol is proven in the UC framework and they also present an efficiency analysis of their construction.

The notion of bounded-rewind security has been considered in previous works with respect to simpler primitives, like witness-indistinguishable proofs, commitment schemes [4, 28, 36] and the mentioned oblivious transfer [10]. In [10] the authors also propose a notion (and instantiation) of a rewindable secure MPC protocol. However, their rewind-secure protocol is only secure against a weaker class of adversaries called *semi-malicious adversaries*. Without getting too technical, such an adversary provides its randomness and inputs to the simulator which can then simulate in a straight-line manner. In our work, we do not have this luxury since we require our construction to be secure against any probabilistic polynomial-time adversarial strategy which creates many additional technical challenges.

2 Preliminaries and Standard Definitions

Notation. 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 $p(|x|)$. Let A and B be two interactive probabilistic algorithms. We denote by $\langle A(\alpha), B(\beta)\rangle(\gamma)$ the distribution of B's output after running on private input β with A using private input α, both running on common input γ. Typically, one of the two algorithms receives 1^λ as an input. A *transcript* of $\langle A(\alpha), B(\beta)\rangle(\gamma)$ consists of the messages exchanged during an execution where A receives a private input α, B receives a private input β and both A and B receive a common input γ. Moreover, we will define the *view* of A (resp. B), denoted by $\mathsf{view}^A_{(A,B)}$ (resp. $\mathsf{view}^B_{(A,B)}$), as the messages it received during the execution of the protocol (A, B), along with its randomness and its input. We say that the transcript τ of an execution $b = \langle A(z), B\rangle(x)$ is *accepting* if $b = 1$. We say that a protocol (A, B) is public coin if B only sends random bits to A. If the randomness is explicit we write $a := A(x; r)$ where x is the input and r is the randomness.

A protocol is defined to be *delayed-input* if it requires the input of the protocol only in the last round of communication.

We assume familiarity with the notion of negligible functions, garbled circuits, CPA encryptions, extractable commitments, public-coin WI proofs and oblivious transfers and refer the reader to the full version for more details.

2.1 Non-Malleable Commitments Scheme

We follow the definition of non-malleable commitments used in [26,38,40] (these definitions are build upon the original definition of Dwork et al. [20]). In the real experiment, the adversary, called man-in-the-middle MIM, interacts with a committer C in the left session, and with a receiver R in the right session. We assume w.l.o.g. that each session has a tag and non-malleability holds only if the tag from the left session is different from the one in the right session.

At the beginning of the experiment, C receives an input v and MIM receives an auxiliary input z, which could contain a priori information about v. For the real experiment, we denote with $\mathsf{MIM}_{\langle C,R\rangle}(\tilde{v}, z)$ the random variable that describes the message that MIM commits to in the right session, jointly with the view of MIM. In the ideal experiment, MIM interacts with a PPT simulator $\mathcal{S}$. There, we denote with $\mathsf{SIM}_{\langle C,R\rangle}(1^\lambda, z)$ the random variable describing the value $\tilde{v}$ that $\mathcal{S}$ committed to and the output view of $\mathcal{S}$. In either of the two experiments, the value $\tilde{v}$ is defined to be $\perp$ if the tags in the left and right session are equal.

Definition 2.1 (Synchronous Non-malleable Commitments). *A 3-round commitment scheme $\langle C, R\rangle$ is said to be synchronous non-malleable if for every PPT* synchronizing adversary[11] MIM*, there exists a PPT simulator $\mathcal{S}$ such that the following ensembles are computationally indistinguishable:*

$$\{\mathsf{MIM}_{\langle C,R\rangle}(\tilde{v}, z)\}_{\lambda\in\mathbb{N},v\in\{0,1\}^\lambda,z\in\{0,1\}^*} \text{ and } \{\mathsf{SIM}_{\langle C,R\rangle}(1^\lambda, z)\}_{\lambda\in\mathbb{N},v\in\{0,1\}^\lambda,z\in\{0,1\}^*}$$

Additionally, we require the following properties to hold for the synchronous non-malleable commitments: 1) *Non-malleability with respect to extraction*: this notion requires the existence of an extractor $\mathtt{Ext}_{\mathsf{NMCom}}$ that is able to extract a message from a well-formed commitment generated by MIM. Moreover, the output distribution of $\mathtt{Ext}_{\mathsf{NMCom}}$ remains the same independently of whether the adversary is receiving honest or simulated commitments. 2) *Last-message*

[11] A synchronizing adversary is an adversary that sends its message for every round before obtaining the honest party's message for the next round.

pseudo-randomness: the last message generated by C is computationally indistinguishable from a random string.

Formal definitions of this properties can be found in the full version. In the work of Choudhuri et al. [10], it is observed that the (synchronous version of the) 3-round non-malleable commitments of [27] satisfies all the mentioned properties.

2.2 Trapdoor Generation Protocol with Bounded Rewind Security

This section is taken almost verbatim from [4, 10] and introduces the notion of trapdoor generation protocols with bounded rewind security.

Syntax. A trapdoor generation protocol $\mathsf{TDGen} = (\mathsf{TDGen}_1, \mathsf{TDGen}_2, \mathsf{TDGen}_3, \mathsf{TDOut}, \mathsf{TDValid}, \mathsf{TDExt})$ is a three round protocol between two parties - a sender (trapdoor generator) S and a receiver R that proceeds as follows:

1. **Round 1 -** $\mathsf{TDGen}_1(\cdot)$**:**
   S computes and sends $\mathsf{td}_1^{S\to R} \leftarrow \mathsf{TDGen}_1(R_S)$ using a random string R_S.
2. **Round 2 -** $\mathsf{TDGen}_2(\cdot)$**:**
   R computes and sends $\mathsf{td}_2^{R\to S} \leftarrow \mathsf{TDGen}_2(\mathsf{td}_1^{S\to R}; R_R)$ using randomness R_R.
3. **Round 3 -** $\mathsf{TDGen}_3(\cdot)$**:**
   S computes and sends $\mathsf{td}_3^{S\to R} \leftarrow \mathsf{TDGen}_3(\mathsf{td}_2^{R\to S}; R_S)$.
4. **Output -** $\mathsf{TDOut}(\cdot)$**:**
   The receiver R outputs $0/1 \leftarrow \mathsf{TDOut}(\mathsf{td}_1^{S\to R}, \mathsf{td}_2^{R\to S}, \mathsf{td}_3^{S\to R})$.
5. **Trapdoor Validation Algorithm -** $\mathsf{TDValid}(\cdot)$**:**
   Taking as an input $(\mathsf{trap}, \mathsf{td}_1^{S\to R})$, output a single bit 0 or 1 that determines whether the value trap is a valid trapdoor corresponding to the message td_1 sent in the first round of the trapdoor generation protocol.

In the remainder of this work, to not overburden the notation, we indicate td_1 to be $\mathsf{td}_1^{S\to R}$, td_2 to be $\mathsf{td}_2^{R\to S}$, and td_3 to be $\mathsf{td}_3^{S\to R}$.

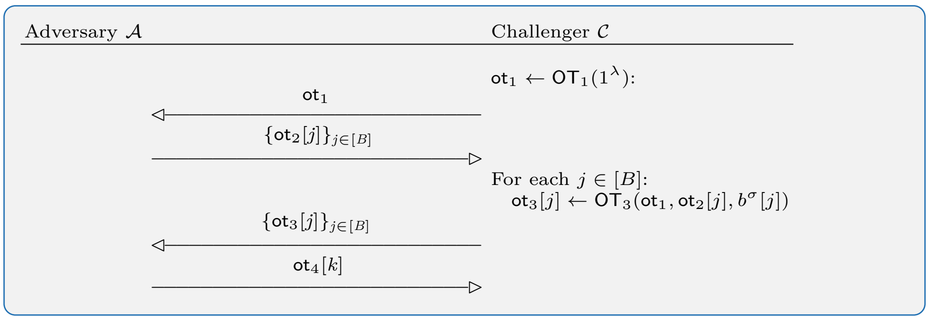
The algorithm $\mathsf{TDValid}$ is public and everyone can verify that trap is a valid trapdoor for a first round message td_1.
Extraction. Furthermore, we require the existence of a PPT extractor algorithm TDExt that, given a set of values[12] $(\mathsf{td}_1, \{\mathsf{td}_2^i, \mathsf{td}_3^i\}_{i=1}^3)$ such that $\mathsf{td}_2^1, \mathsf{td}_2^2, \mathsf{td}_2^3$ are distinct and $\mathsf{TDOut}(\mathsf{td}_1, \mathsf{td}_2^i, \mathsf{td}_3^i) = 1$ for all $i \in [3]$, outputs a trapdoor trap such that $\mathsf{TDValid}(\mathsf{trap}, \mathsf{td}_1) = 1$.
1-Rewinding Security. Roughly speaking, if a trapdoor generation protocol is 1-rewind secure then no cheating PPT receiver $R^\star$ can learn a valid trapdoor even when $R^\star$ queries S on two (possibly adaptive) different second-round messages, thereby receiving two different third round responses from the sender. The formal definition of this notion can be found in the full version.

[12] These values can be obtained from the malicious sender via an expected PPT rewinding procedure. The expected PPT simulator in our applications performs the necessary rewindings and then inputs these values to the extractor TDExt.

3 Rewind-Secure OT and MPC

We assume familiarity with the standard notions of multi-party computation secure with unanimous and identifiable abort under black-box simulation in the plain model. We refer to the full version for more details. In this section, we introduce the definitions of two rewind-secure primitives, namely oblivious transfer (OT) and MPC. We start with the definition of our new notion of special rewindable OT. Afterwards, we define what it means for an MPC protocol to be rewindable secure.

Definition 3.1 (Special B-Rewindable OT Security). *Let* $\mathsf{OT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3, \mathsf{OT}_4)$ *be an OT protocol, 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_k *and* E^1_k *(where* E^σ_k *is defined below) 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* $b^0[k] = b^1[k]$.


Adversary $\mathcal{A}$
Challenger $\mathcal{C}$
$\mathsf{ot}_1 \leftarrow \mathsf{OT}_1(1^\lambda)$:
ot_1
$\{\mathsf{ot}_2[j]\}_{j\in[B]}$
For each $j \in [B]$:
$\mathsf{ot}_3[j] \leftarrow \mathsf{OT}_3(\mathsf{ot}_1, \mathsf{ot}_2[j], b^\sigma[j])$
$\{\mathsf{ot}_3[j]\}_{j\in[B]}$
$\mathsf{ot}_4[k]$

We note that this definition is equal to the one proposed in [10] except for the fact that we require the adversary to pick the same input in the k-th slot.

Definition 3.2 (Bounded Rewind-Secure MPC with unanimous abort). *A 4-round MPC protocol* MPC *for f is a tuple of deterministic polynomial-time algorithms* $\mathsf{MPC} = \{(\mathsf{Next}^1_i, \mathsf{Next}^2_i, \mathsf{Next}^3_i, \mathsf{Next}^4_i, \mathsf{output}_i)\}_{i\in[n]}$ *(where the algorithms are defined as in the standard definition of MPC:*

Similar to the standard security definition of MPC, we define the real-world and ideal-world execution.

Ideal Computation. *Let* $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ *be an n-party function and let* $\mathcal{I} \subset [n]$*, of size at most* $n-1$*, be the set of indices of the corrupt parties. Then, the joint ideal execution of f under* $(\mathcal{S}, \mathcal{I})$ *on input vector* $x = (x_1, \ldots, x_n)$*, auxiliary input* aux *and security parameter* λ*, denoted by* $\mathtt{IDEAL}^{\mathsf{un\text{-}abort}}_{f,\mathcal{I},\mathcal{S}(\mathsf{aux})}(x, \lambda)$*, is defined as in the standard definition of MPC.*

Real Execution. *Let* $\Pi = (P_1, \ldots, P_n)$ *be an n-party 4-round MPC protocol and let* $\mathcal{I} \subseteq [n]$*, of size at most* $n-1$*, denote the set of indices of the parties corrupted by* $\mathcal{A}$*. The joint execution of* Π *under* $(\mathcal{A}, \mathcal{I})$ *in the real world, on input vector* $x = (x_1, \ldots, x_n)$*, auxiliary input* aux *and security parameter* λ*, denoted by* $\mathtt{REAL}_{\Pi,\mathcal{I},\mathcal{A}(\mathsf{aux})}(x, \lambda)$*, is defined as the output vector of* $P_1, \ldots, P_n$ *and* $\mathcal{A}(\mathsf{aux})$

resulting from the following 4*-round protocol interaction. Let* $\mathcal{H}$ *denote the set of indices of honest parties* $\mathcal{H} = [n] \setminus \mathcal{I}$.

- *Interaction in Round 1:* $\mathcal{A}$ *receives* $\{\mathsf{msg}_j^1 = \mathsf{Next}_j(1^\lambda, x_j, \rho_j)\}_{j\in\mathcal{H}}$ *and sends messages* $\{\mathsf{msg'}_i^1\}_{i\in\mathcal{I}}$ *of its choice. Let* $\overline{\mathsf{msg}}^{<2} = \{\{\mathsf{msg}_j^1\}_{j\in\mathcal{H}}, \{\mathsf{msg'}_i^1\}_{i\in\mathcal{I}}\}$.
- *Interaction in Round 2 and 3 with B rewinds:*
  - $\mathcal{A}$ *is given* $\{\mathsf{msg}_j^2 = \mathsf{Next}_j^2(1^\lambda, x_j, \rho_j, \overline{\mathsf{msg}}^{<2})\}_{j\in\mathcal{H}}$.
  - $\mathcal{A}$ *chooses B second-round messages, namely* $\{\mathsf{msg'}_i^2[k]\}_{i\in\mathcal{I},k\in[B]}$.
  - $\mathcal{A}$ *is given* $\{\mathsf{msg}_j^3[k] = \mathsf{Next}_j^3(1^\lambda, x_j, \rho_j, \overline{\mathsf{msg}}^{<3}[k])\}_{j\in\mathcal{H},k\in[B]}$, *where* $\overline{\mathsf{msg}}^{<3}[k] = \{\overline{\mathsf{msg}}^{<2}, \{\mathsf{msg}_j^2\}_{j\in\mathcal{H}}, \{\mathsf{msg'}_i^2[k]\}_{i\in\mathcal{I}}\}$.
  - $\mathcal{A}$ *sends third-round message* $\{\mathsf{msg'}_i^3\}_{i\in\mathcal{I}}$ *of its choice.*
  - *Let* $\overline{\mathsf{msg}}^{<4} = \{\overline{\mathsf{msg}}^{<3}[1], \{\mathsf{msg}_j^3[1]\}_{j\in\mathcal{H}}, \{\mathsf{msg'}_i^3\}_{i\in\mathcal{I}}\}$.
- *Interaction in Round 4:* $\mathcal{A}$ *is given fourth-round messages* $\{\mathsf{msg}_j^4 = \mathsf{Next}_j^4(1^\lambda, x_j, \rho_j, \overline{\mathsf{msg}}^{<4})\}_{j\in\mathcal{H}}$. $\mathcal{A}$ *sends fourth-round messages* $\{\mathsf{msg'}_i^4\}_{i\in\mathcal{I}}$ *of its choice.*

$\mathtt{REAL}_{\mathcal{I},\mathcal{A}(\mathsf{aux})}(\overline{x}, \lambda)$ *is defined as* $(\overline{y}_{\mathcal{H}}, z)$, *where* $\overline{y}_{\mathcal{H}}$ *is the vector of outputs of the honest parties while z is the output of the adversary.*

Security Definition. *Let* $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ *be an n-party function. A protocol* Π *securely computes the function f with unanimous abort and bounded B-rewind security if for every PPT real-world adversary* $\mathcal{A}$ *there exists a PPT simulator* $\mathcal{S}$ *such that for every* $\mathcal{I} \subset [n]$ *of size at most* $n-1$, *the following ensembles are computationally indistinguishable:*

$$\left\{\mathtt{REAL}_{\Pi,\mathcal{I},\mathcal{A}(\mathsf{aux})}(x,\lambda)\right\}_{x\in(\{0,1\}^*)^n,\lambda\in\mathbb{N}} \text{ and } \left\{\mathtt{IDEAL}^{\mathsf{un\text{-}abort}}_{f,\mathcal{I},\mathcal{S}(\mathsf{aux})}(x,\lambda)\right\}_{x\in(\{0,1\}^*)^n,\lambda\in\mathbb{N}}.$$

4 From MPC with Unanimous Abort to B-rewindable MPC with Unanimous Abort

In this section, we present a compiler that makes a four-round MPC protocol Π_{MPC} secure with unanimous abort in the plain model bounded-rewind-secure, resulting in the protocol Π_{rmpc}, while preserving all its other security properties. We begin with a high-level overview of the compiler and establish some notation for simplicity. Let msg_i^r denote the message broadcast by P_i in Round r ($r \in [4]$) of Π_{MPC}.

The two-party case. For simplicity, we start by considering the 2-party case, where one of the parties, here P_2, is corrupted. To make Π_{MPC} bounded-rewind-secure, we need to ensure that security is maintained even if P_2 receives a set of multiple third-round messages from P_1 (namely, msg_1^3) as a response to its chosen set of second-round messages (namely, msg_2^2).

In our protocol the party P_i ($i \in [2]$) computes a garbled circuit $\mathtt{GC}_i$ that has hard-coded inside its input x_i, randomness r_i and the protocol transcript of Π_{MPC} until Round 2 i.e. $\{\mathsf{msg}_j^1, \mathsf{msg}_j^2\}_{j\in[2]}$ and takes as input the set of third-round messages $\{\mathsf{msg}_j^3\}_{j\in[2]}$ and outputs P_i's fourth-round message i.e. msg_i^4. In the last round of Π_{rmpc}, these garbled circuits are then evaluated to obtain the fourth-round messages of Π_{MPC}. For the evaluation of these garbled circuits, we need the parties to be able to obtain the labels corresponding to $\{\mathsf{msg}_j^3\}_{j\in[2]}$. For this purpose we use the rely on an oblivious transfer OT protocol.[13] In the above context, P_1 does not send msg_1^3 directly in Round 3 of Π_{rmpc} but instead participates in an OT instance as an OT receiver with input msg_1^3; while P_2 participates as the sender using as its input the labels of $\mathtt{GC}_2$ that were used to encode the input msg_1^3. Similarly, there would be another OT instance with P_2 as the receiver and P_1 as the sender for the labels of msg_2^3 corresponding to $\mathtt{GC}_1$. It is now evident that the parties can proceed to evaluate the garbled circuits, obtain the fourth-round messages of Π_{MPC} and compute the output. Intuitively, the above approach helps to achieve rewind-security because, in the security game, the honest party P_1 has to send multiple third-round messages (in round 3 of Π_{rmpc}) which, in our protocol, contain msg_1^3 messages under the hood of OT and are thereby 'hidden' from the adversary. In the fourth-round only one among these third-round messages is 'opened' to the adversary which effectively reduces the security to a single execution of Π_{MPC}.

The multi-party case. A natural extension of the above 2-party approach to the multi-party case would be to let each pair of parties, P_i and P_j, engage in an OT instance (say $\mathsf{OT}^{j,i}$) as an OT receiver and sender respectively to retrieve labels of msg_i^3 corresponding to $\mathtt{GC}_j$ (which would output the fourth-round message of P_j). However, unlike the 2-party case, a party, for example, P_i is not able to obtain the labels for all the inputs of $\mathtt{GC}_j$, and therefore cannot evaluate $\mathtt{GC}_j$. In more detail, P_i would not have access to the labels corresponding to the input msg_k^3 in $\mathtt{GC}_j$ (where $k, j \neq i$). To enable P_i (and everyone else) to recover these labels, we make P_k reveal the OT randomness that is used as an OT receiver during the instance $\mathsf{OT}^{j,k}$ so that everyone can learn the output of this OT (i.e. the labels of $\mathtt{GC}_j$ corresponding to msg_k^3). Note that this randomness can be safely revealed because the adversary learns the OT receiver's input msg_k^3 also in the protocol Π_{MPC} which is secure with unanimous abort. In light of the above, we define the security notion of OT with rewindable security against a malicious sender. This notion is a slightly modified variant of the one used in [10] and their construction can be easily adapted with minor tweaks to satisfy our notion. This rewind-secure OT construction also satisfies *public verifiability*, enabling all the parties to check the correctness of the OT receiver messages, for all pairwise instances of the OT, by just checking the transcripts.

Next, we observe that the above approach of using pairwise OTs is vulnerable to the following attack by a potentially corrupted party P_i : P_i could use different third-round messages (i.e. msg_i^3) as its input across the n OT instances where it acts as an OT receiver (one instance for every other party as the sender). This behaviour violates the security of the underlying MPC since it allows the adversary to recover the fourth-round messages of honest parties (via evaluation of their respective garbled circuits) computed with respect to different third-round messages msg_i^3. Note that in the underlying protocol Π_{MPC} the adversary cannot

[13] For the wires corresponding to their own third-round message (i.e. msg_i^3 in $\mathtt{GC}_i$), the labels can be broadcasted directly in the last round.

launch this attack since honest parties compute their fourth-round message with respect to the same msg_i^3 (which is broadcast in round 3 of Π_{MPC}).

The crux of the above issue is that the labels of $\mathtt{GC}_j$ corresponding to msg_i^3 are tied to a single instance of OT, i.e. the one between P_i and P_j. To resolve this, we break this one-to-one dependency such that the labels of $\mathtt{GC}_j$ corresponding to msg_i^3 are obtained in a distributed manner across all n OTs where P_i acts as a receiver. For simplicity, assume that msg_i^3 contains only a single bit $b \in \{0, 1\}$ and corresponds to wire w in each of the n garbled circuits. Each garbler P_j additively shares the labels of wire w of $\mathtt{GC}_j$ among the parties for $b \in \{0, 1\}$. Now, each P_k has an additive share for each of the n garbled circuits corresponding to wire w and each bit $b \in \{0, 1\}$. Accordingly, the OT instances between P_i, acting as the receiver, (who participates with the actual value of bit b as input) and P_k, acting as the sender, would now involve P_k participating with the two tuples of n additive shares as its input, where the first tuple comprises of the n additive shares for $b = 0$, while the other tuple contains the n additive shares for $b = 1$. The above technique ensures that if P_i participates with inconsistent inputs b across its instances as an OT receiver then neither the label for $b = 0$ nor for $b = 1$ will be recovered for any honest party's garbled circuit. This is due to the fact that the OT instances with a subset of honest parties as OT senders would output additive shares corresponding to 0, while the others would output additive shares corresponding to 1; which is insufficient to reconstruct either of the labels. The transfer of the additive shares is done using public-key encryption i.e. by encrypting the relevant share using the public key of the intended recipient. This allows us to maintain the property that all messages in Π_{rmpc} are sent over a broadcast channel.

Looking ahead, this is useful to achieve identifiable abort security as it allows the parties to give a corresponding proof of correctness for these messages in such a way that it can be verified by everyone. In our final construction, however, the relevant additive shares of the garbled circuits are not used directly in the OT instances. Instead, the OT senders encrypt each of their tuple of n additive shares using one-time pads, broadcast these encryptions and use the corresponding one-time pad keys as inputs to the OT. Note that if the additive shares were used directly, some of the components of an honest sender's input (corresponding to the additive shares given by a corrupt garbler) are adversarially chosen. However, the above described modification using one-time pads allows us to rely on standard OT security where an honest sender's input is not adversarially chosen. This completes the high-level description of our compiler.

Lastly, we highlight an important aspect related to the security of the above described bounded-rewind-secure MPC construction. Since we allow the adversary to proceed to the evaluation of the garbled circuits and obtain the output only if it used consistent third-round messages in all the OT instances where it participated as a receiver, we require the property of 'simultaneous extractability' from the rewind-secure OT. In more detail, consider multiple OT instances running in parallel where the receiver is corrupted and the sender is honest. We require that the simulator of the OT should be able to extract the input of the malicious receivers in the same rewinding thread for multiple OT instances. This is needed to check if the adversary used consistent inputs on behalf of the same malicious receiver or not, as the latter would result in abort. We show in Section 6 that the modified variant of the rewind-secure OT of [10] satisfies this property of simultaneous extractability.

We now proceed to the formal description of our compiler.

The compiler makes use of the following tools:

– A four-4 MPC protocol Π_{MPC} with unanimous abort security represented by the set of functions $\{(\mathsf{Next}^1_i, \mathsf{Next}^2_i, \mathsf{Next}^3_i, \mathsf{Next}^4_i, \mathsf{output}_i)\}_{i\in[n]}$, where all the messages are assumed to be sent over a broadcast channel.
– A garbling scheme $(\mathsf{garble}, \mathsf{eval}, \mathsf{simGC})$ that is assumed to satisfy properties of privacy (a set of labels, together with the garbled circuit, reveals nothing about the input the labels correspond to), correctness (the correct evaluation of the garbled circuit matches the evaluation of the plain circuit), authenticity of input labels (it is not possible to 'forge' a different set of valid input labels from a set of valid input labels) and partial evaluation resiliency (unless at least one label corresponding to every bit is obtained, nothing about the output is revealed). We defer details of these notions to the full version.
– A delayed-input OT protocol, instantiated by the construction in Section 6, denoted as a sequence of algorithms $(\mathsf{rOT}_1, \mathsf{rOT}_2, \mathsf{rOT}_3, \mathsf{rOT}_4, \mathsf{rOT}_5, \mathsf{rOT}_6)$, where rOT_r $(r \in [5])$ denotes the algorithm to compute the r-th round messages and rOT_6 denotes the algorithm for the output computation. The OT protocol satisfies special 2-B rewindable security and sender simulatability.
– A CPA-secure public-key encryption scheme $\mathsf{PKE} = (\mathsf{keygen}, \mathsf{enc}, \mathsf{dec})$.

Figure 4.1: Π_{rmpc}

Notation. - Let Circuit $\mathsf{C}_{i,x,\rho,\overline{\mathsf{msg}}^{<3}}(\mathsf{msg}^3_1, \mathsf{msg}^3_2, \ldots, \mathsf{msg}^3_n)$ denote the boolean circuit with hard-wired values i, x, ρ and the transcript $\overline{\mathsf{msg}}^{<3}$ of the first two rounds of an execution of Π_{MPC} that upon receiving n inputs $\mathsf{msg}^3_1, \mathsf{msg}^3_2, \ldots, \mathsf{msg}^3_n$ (i.e. the third-round broadcast messages of the execution of Π_{MPC}) computes $\mathsf{Next}^4_i(x, \rho, \overline{\mathsf{msg}}^{<4} = \{\overline{\mathsf{msg}}^{<3}, \mathsf{msg}^3_1, \mathsf{msg}^3_2, \ldots, \mathsf{msg}^3_n\})$. For simplicity, we assume that each third-round broadcast message is ℓ bits long – so the circuit has $\mathsf{L} = n\ell$ input bits.

- Let $\mathsf{OT}^{j,k}$ denote an instance of a B-rewind secure OT (denoted as $(\mathsf{rOT}_1, \mathsf{rOT}_2, \mathsf{rOT}_3, \mathsf{rOT}_4, \mathsf{rOT}_5, \mathsf{rOT}_6)$) where P_j acts as the sender and P_k acts as the receiver.

Private Input. P_i has private input $x_i \in \{0,1\}^\lambda$ and randomness ρ_i.

Output. $y = f(x_1, \ldots, x_n)$ or $\bot$.

Round 1. Each P_i does the following:

1. Run the setup of the PKE scheme as $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{keygen}(1^\lambda; R_{\mathsf{PKE}})$.
2. Compute the first-round message of Π_{MPC} as $\mathsf{msg}^1_i \leftarrow \mathsf{Next}^1_i(x_i; \rho_i)$.

3. Compute the first-round OT message as the receiver – i.e. for each $j \in [n]$ corresponding to the instance $\mathsf{OT}^{j,i}$ (where P_i acts as the receiver), sample randomness $R^1_{j,i}$ and compute $\mathsf{rot}^{j,i}_1 \leftarrow \mathsf{rOT}_1(1^\lambda; R^1_{j,i})$.
4. Broadcast $\big(\mathsf{pk}_i, \mathsf{msg}^1_i, \{\mathsf{rot}^{j,i}_1\}_{j\in[n]}\big)$.

Round 2. Each P_i does the following:

1. If any party aborts in the previous round, honest parties output $\perp$. [a]
2. Compute the second-round message of Π_{MPC} as $\mathsf{msg}^2_i \leftarrow \mathsf{Next}^2_i(x_i, \rho_i, \overline{\mathsf{msg}}^{<2})$, where $\overline{\mathsf{msg}}^{<2} = \{\mathsf{msg}^1_j\}_{j\in[n]}$.
3. Compute the second-round OT message as the sender – for each $j \in [n]$ corresponding to the instance $\mathsf{OT}^{i,j}$ (where P_i acts as the sender), sample randomness $S_{i,j}$ and compute $\mathsf{rot}^{i,j}_2 \leftarrow \mathsf{rOT}_2(\mathsf{rot}^{i,j}_1; S_{i,j})$.
4. Broadcast $\big(\mathsf{msg}^2_i, \{\mathsf{rot}^{i,j}_2\}_{j\in[n]}\big)$.

Round 3. Each P_i does the following:

1. Compute the garbled circuit as $(\mathtt{GC}_i, \vec{K}_i) \leftarrow \mathsf{garble}(\mathsf{C}_{i,x,\rho,\overline{\mathsf{msg}}^{<3}}, 1^\lambda; R_{\mathtt{GC}})$, where $\vec{K}_i$ denotes the set of labels $\{\mathsf{K}^{(0)}_{i,\alpha}, \mathsf{K}^{(1)}_{i,\alpha}\}_{\alpha\in[\mathsf{L}]}$.
2. For each $\alpha \in [\mathsf{L}]$ and $b \in \{0,1\}$, compute an additive sharing $(\mathsf{K}^{(b)}_{i,\alpha,1}, \mathsf{K}^{(b)}_{i,\alpha,2}, \ldots, \mathsf{K}^{(b)}_{i,\alpha,n})$ of the label $\mathsf{K}^{(b)}_{i,\alpha}$.
3. For each $\alpha \in [\mathsf{L}]$, $b \in \{0,1\}$ and $j \in [n] \setminus \{i\}$, compute the ciphertexts $\mathsf{ct}^{(b)}_{i,\alpha,j} \leftarrow \mathsf{enc}(\mathsf{pk}_j, \mathsf{K}^{(b)}_{i,\alpha,j})$.
4. Compute the third-round message of Π_{MPC} as $\mathsf{msg}^3_i \leftarrow \mathsf{Next}^3_i(x_i, \rho_i, \overline{\mathsf{msg}}^{<3})$, where $\overline{\mathsf{msg}}^{<3} = \{\mathsf{msg}^1_j, \mathsf{msg}^2_j\}_{j\in[n]}$.
5. Compute the third-round OT message as the receiver using as an input the string msg^3_i – for each $j \in [n]$ corresponding to the OT instance $\mathsf{OT}^{j,i}$, sample randomness $R^3_{j,i}$ and run $\mathsf{rot}^{j,i}_3 \leftarrow \mathsf{rOT}_3(\mathsf{msg}^3_i, \mathsf{rot}^{j,i}_1, \mathsf{rot}^{j,i}_2; R^3_{j,i})$.
6. Broadcast $\big(\{\mathsf{ct}^{(b)}_{i,\alpha,j}\}_{\alpha\in[\mathsf{L}], j\in[n]\setminus i, b\in\{0,1\}}, \{\mathsf{rot}^{j,i}_3\}_{j\in[n]}\big)$.

Round 4. Each P_i does the following:

1. For each $\alpha \in [\mathsf{L}], b \in \{0,1\}$ and $j \in [n] \setminus \{i\}$, compute $\mathsf{K}^{(b)}_{j,\alpha,i} \leftarrow \mathsf{dec}(\mathsf{sk}_i, \mathsf{ct}^{(b)}_{j,\alpha,i})$.
2. For each $j, k \in [n]$, check the correctness of $\mathsf{rot}^{j,k}_3$ sent by P_k (this is possible due to public verifiability of the OT). If the check does not pass, broadcast '`abort`' and output $\perp$, else, continue.
3. // Recall that the input wires of each $\mathtt{GC}_k$ ($k \in [n]$) at indices $[(j-1)\ell+1, j\ell]$ correspond to msg^3_j and $\mathsf{K}^{(b)}_{k,\alpha,i}$ denotes P_i's additive share of the label of $\mathtt{GC}_k$ corresponding to index α and bit b.

   For each $j \in [n]$ and $\beta \in [\ell]$ – let $m^{(0)}_{j,\beta,i} = (\mathsf{K}^{(0)}_{1,(j-1)\ell+\beta,i}, \ldots, \mathsf{K}^{(0)}_{n,(j-1)\ell+\beta,i})$ and $m^{(1)}_{j,\beta,i} = (\mathsf{K}^{(1)}_{1,(j-1)\ell+\beta,i}, \ldots, \mathsf{K}^{(1)}_{n,(j-1)\ell+\beta,i})$. Sample random strings $q^{(0)}_{j,\beta,i}$ and $q^{(1)}_{j,\beta,i}$ (to be used as one-time pad keys) and compute $M^{(0)}_{j,\beta,i} = m^{(0)}_{j,\beta,i} + q^{(0)}_{j,\beta,i}$ and $M^{(1)}_{j,\beta,i} = m^{(1)}_{j,\beta,i} + q^{(1)}_{j,\beta,i}$.

4. For each $j \in [n]$ corresponding to the OT instance $\mathsf{OT}^{i,j}$ (where P_j participated as a receiver with input msg_j^3) – Compute the fourth-round OT message as the sender as follows: run $\mathsf{rot}_4^{i,j} \leftarrow \mathsf{rOT}_4\Big((q_{j,1,i}^{(0)}, q_{j,1,i}^{(1)}), (q_{j,2,i}^{(0)}, q_{j,2,i}^{(1)}), \ldots, (q_{j,\ell,i}^{(0)}, q_{j,\ell,i}^{(1)}), \mathsf{rot}_1^{i,j}, \mathsf{rot}_2^{i,j}, \mathsf{rot}_3^{i,j}; S_{i,j}\Big)$.
5. For all OT instances $\mathsf{OT}^{j,i}$ where P_i participated as the receiver, compute $\mathsf{rot}_5^{j,i} \leftarrow \mathsf{rOT}_5\Big(R_{j,i}^1, R_{j,i}^3\Big)$ for all $j \in [n]$.
6. Broadcast $(\mathtt{GC}_i, \mathsf{msg}_i^3, \{\mathsf{rot}_4^{i,j}\}_{j\in[n]}, \{\mathsf{rot}_5^{j,i}\}_{j\in[n]}, \{M_{j,\beta,i}^{(0)}, M_{j,\beta,i}^{(1)}\}_{j\in[n],\beta\in[\ell]})$.

Output Computation. Each P_i does the following:
1. If any party broadcasted '`abort`' in Round 4, output $\perp$.
2. Compute the output of each OT instance where P_j acts as the receiver and P_k acts as the sender as follows (where $j, k \in [n]$):
   - Compute $(q_{j,1,k}, q_{j,2,k}, \ldots, q_{j,\ell,k}) \leftarrow \mathsf{rOT}_6(\mathsf{rot}_1^{k,j}, \mathsf{rot}_2^{k,j}, \mathsf{rot}_3^{k,j}, \mathsf{rot}_4^{k,j}, \mathsf{rot}_5^{k,j}, \mathsf{msg}_j^3)$.
   - Let msg_j^3 be the third-round message of Π_{MPC} broadcast by P_j in Round 4. Parse msg_j^3 as $\mathsf{msg}_j^3 = b_{j,1}||b_{j,2}|| \ldots b_{j,\ell}$.
   - For each $\beta \in [\ell]$, set $\{\mathsf{K}_{1,(j-1)\ell+\beta,k}, , \ldots, \mathsf{K}_{n,(j-1)\ell+\beta,k}\} = q_{j,\beta,k} \oplus M_{j,\beta,k}^{(b_{j,\beta})}$, where $M_{j,\beta,k}^{(b_{j,\beta})}$ was broadcast by P_k in Round 4.
3. For each garbled circuit $\mathtt{GC}_j$ ($j \in [n]$) – compute $\mathsf{K}_{j,\alpha} = \sum_{k=1}^{n} \mathsf{K}_{j,\alpha,k}$ for each $\alpha \in [\mathsf{L}]$.
4. For each $\mathtt{GC}_j$ ($j \in [n]$), compute $\mathsf{msg}_j^4 \leftarrow \mathsf{eval}(\mathtt{GC}_j, K_{j,1}, \ldots, K_{j,\mathsf{L}})$.
5. Output $y_i \leftarrow \mathsf{output}_i(x_i, \rho_i, \{\mathsf{msg}_j^1, \mathsf{msg}_j^2, \mathsf{msg}_j^3, \mathsf{msg}_j^4\}_{j\in[n]})$.

[a] we assume that honest parties execute this step in the beginning of each round.

Theorem 4.1. *Assume the existence of a* 4*-round MPC protocol with unanimous abort security against dishonest majority, a CPA-secure public key encryption scheme, a garbling scheme that is assumed to satisfy properties of privacy, correctness, authenticity of input labels and partial evaluation resiliency and a* 5*-round delayed-input oblivious transfer protocol (described in Section 6) satisfying special* 2*-B rewindable security and sender simulatability. Then,* Π_{rmpc} *is a* 4*-round B-rewindable secure MPC with unanimous abort against dishonest majority in the plain model.*

The above construction can be built based on trapdoor permutations (we refer to the full version for details on the instantiations of each of the building blocks). The formal proof of Theorem 4.1 can be found in the full version.

5 Our Construction: MPC with Identifiable Abort

In this section we present the four-round MPC protocol secure with identifiable abort. The idea of our construction is to let every participating party prove, during the execution of the MPC protocol, that it is generating all of its messages according to the protocol description. To prove the correctness of the generated messages, each party, in the last round, executes a zap (i.e., a two-round public-coin WI proof) with every other participating party. These zaps prove that either all the messages of the MPC protocol are generated correctly or the party has, earlier in the protocol execution, generated a non-malleable commitment with respect to a trapdoor, that has been generated using a trapdoor generation protocol later in the protocol execution. Both, the trapdoor generation protocol and the non-malleable commitment scheme are also executed in parallel to the MPC protocol.

To allow the simulator in the security proof of this construction to generate a valid zap it needs to prove the second part of the relation, i.e. that the non-malleable commitment is a commitment to the trapdoor generated using the trapdoor generation protocol. To create such a commitment, the simulator needs to rewind the overall protocol to extract the trapdoor from the trapdoor generation protocol. To guarantee that during these rewinds the underlying MPC protocol is preserved, we can rely on its rewindable security. After the trapdoor is extracted, the simulator can commit to it in the non-malleable commitment and, finally, finishes the execution of the zap. We need to require the commitment scheme to be non-malleable to prevent an adversary from malleability attacks. An adversary could, for example, if these commitments were malleable, maul one of them during the simulation and use it to create its own commitment to the trapdoor of the trapdoor generation protocol and use it to provide an accepting zap even though it did not behave accordingly to the protocol description. In Figure 5.1 we provide the formal description of our protocol Π^{ID}, for which we make use of the following tools.

- A public-coin perfectly correct trapdoor generation protocol $\mathsf{TDGen} = (\mathsf{TDGen}_1, \mathsf{TDGen}_2, \mathsf{TDGen}_3, \mathsf{TDOut}, \mathsf{TDValid}, \mathsf{TDExt})$.
- A perfectly correct 3-round special non-malleable commitment scheme $\mathsf{NMCom} = (C, R)$.
- A perfectly correct 4-round MPC protocol that is 3-rewindable secure with unanimous abort Π. W.l.o.g. we assume that whenever a party aborts in Π (i.e., its next message function outputs $\perp$) then the party keeps interacting with the other parties by sending $\perp$ anytime that it is suppose to send a message for Π and replace its output with $\perp$. Moreover, if a party receives $\perp$ from any other party, it will replace any message of Π (as well as its output) with $\perp$. The construction of Π is described in Section 4 and we argue that it satisfies perfect correctness in the full version.
- A perfectly correct 2-round public coin WI proof $\mathsf{RWI} = (\mathcal{P}, \mathcal{V})$ for the NP-language L characterized by the relation R specified below (we denote statements and witnesses as $\mathtt{st}$ and $\mathtt{w}$, respectively).

$\mathtt{st} := \left(\overline{\mathsf{msg}}^{<5}, \{\mathsf{msg}_\ell\}_{\ell\in[4]}, \{\mathsf{nmc}_\ell\}_{\ell\in[3]}, \mathsf{td}_1, r\right)$ and $\mathtt{w} := (x, R, \tilde{r}, R_{\mathsf{NMCom}})$

$R(\mathtt{st}, \mathtt{w}) = 1$ if *either* of the following conditions is satisfied:

1. **Honest:** for every $\ell \leq 4$, msg_ℓ is an honestly computed ℓ^{th} round message in the protocol Π w.r.t. input x, randomness R and the first $(\ell - 1)$ round protocol transcript $\overline{\mathsf{msg}}^{<5}$.
2. **Trapdoor:** $\{\mathsf{nmc}_\ell\}_{\ell\in[3]}$ is an honest transcript of NMCom w.r.t. input $\tilde{r}$ and randomness R_{NMCom} (AND) $\mathsf{trap} = r \oplus \tilde{r}$ is a valid trapdoor w.r.t. td_1

We also require the domain of the messages of the receivers/verifier of TDGen, NMCom and RWI to be $\{0,1\}^\lambda$.

Figure 5.1: Π^{ID}

In each round if a set of parties stops replying then all the honest parties stop and output $(\perp, i)$, where P_i is the party with the smallest index that did not reply.

Round 1. P_i computes and broadcasts the *first* round messages of the following protocols:

1. Rewindable secure MPC Π: $\mathsf{msg}_{1,i} \leftarrow \mathsf{Next}_{1,i}(1^\lambda, x_i, R_i, \perp)$.
2. Sender message of TDGen: $\mathsf{td}_{1,i} \leftarrow \mathsf{TDGen}_1\,(R_{\mathsf{td},i})$.

For every $j \neq i$:

3. Sender message of the non-malleable commitment scheme $\mathsf{NMCom}_1^{i\to j} \leftarrow \mathsf{NMCom}_1(\tilde{r}^{i\to j}, R_{\mathsf{NMCom}}^{i\to j})$ where $\tilde{r}^{i\to j} \leftarrow \{0,1\}^\lambda$.

Round 2. P_i computes and broadcasts the *second* round messages of the following protocols:

1. MPC Π: $\mathsf{msg}_{2,i} \leftarrow \mathsf{Next}_{2,i}(1^\lambda, x_i, R_i, \overline{\mathsf{msg}}^{<2})$.

For every $j \neq i$:

2. Receiver message of TDGen: $\mathsf{td}_2^{i\to j} \leftarrow \mathsf{TDGen}_2(\mathsf{td}_{1,j})$.
3. Receiver message of the non-malleable commitment scheme $\mathsf{NMCom}_2^{j\to i} \leftarrow \mathsf{NMCom}_2(\mathsf{nmc}_1^{j\to i})$.

Round 3. P_i computes and broadcasts the following messages of the following protocols:

1. Third round of Π: $\mathsf{msg}_{3,i} \leftarrow \mathsf{Next}_{3,i}(1^\lambda, x_i, R_i, \overline{\mathsf{msg}}^{<3})$
2. The third round of TDGen: set $\mathsf{td}_{2,i} = \mathsf{td}_2^{1\to i}||\ldots||\mathsf{td}_2^{n\to i}$ where $\mathsf{td}_2^{i\to i} = \perp$. Compute $\mathsf{td}_{3,i} \leftarrow \mathsf{TDGen}_3(\mathsf{td}_{2,i})$.

   For every $j \neq i$:

   (a) The third round of NMCom: $\mathsf{NMCom}_3^{i\to j} \leftarrow \mathsf{NMCom}_3(\mathsf{nmc}_2^{i\to j}, \tilde{r}^{i\to j}; R_{\mathsf{NMCom}}^{i\to j})$.

   (b) The first round of RWI: $\mathsf{zap}_1^{j\to i} \leftarrow \{0,1\}^\lambda$.

Round 4. P_i does the following: If $\exists j \neq i$ such that $\mathsf{TDValid}(\mathsf{td}_{1,j}, \mathsf{td}_{2,j}, \mathsf{td}_{3,j}) \neq 1$ then output $(\mathtt{abort}, j)$ and **stop** else compute and broadcast the following messages

// where $\mathsf{td}_{2,j} := (\mathsf{td}_2^{1\to j}||\cdots||\mathsf{td}_2^{n\to j})$.

1. Fourth round message of the MPC protocol Π: $\mathsf{msg}_{4,i} \leftarrow \mathsf{Next}_{4,i}(1^\lambda, x_i, R_i, \overline{\mathsf{msg}}^{<4})$

For every $j \neq i$

2. A random value $r^{i\to j} \leftarrow \{0,1\}^\lambda$.
3. The second round of RWI: Define $\mathtt{st} := \Big(\overline{\mathsf{msg}}^{<5}, \big\{\mathsf{msg}_{\ell,i}\big\}_{\ell\in[4]}, \{\mathsf{nmc}_\ell^{i\to j}\}_{\ell\in[3]}, \mathsf{td}_{1,j}, r^{i\to j}\Big)$ and $\mathtt{w} := (x_i, R_i, \perp, \perp)$ and compute $\mathsf{zap}_2^{i\to j} \leftarrow \mathcal{P}(\mathtt{st}, \mathtt{w}, \mathsf{zap}_1^{i\to j})$.

Output Computation P_i computes the following:

1. If $\exists j \neq i$ and k, s.t. $\mathcal{V}(\mathtt{st}, \mathsf{zap}_2^{j\to k}, \mathsf{zap}_1^{j\to k}, \mathtt{st}) = 0$ where $\mathtt{st} := \Big(\overline{\mathsf{msg}}^{<5}, \big\{\mathsf{msg}_{\ell,j}\big\}_{\ell\in[4]}, \{\mathsf{nmc}_\ell^{j\to k}\}_{\ell\in[3]}, \mathsf{td}_{1,k}, r_j\Big)$, output j and **stop**.
2. Output $\mathsf{output}(1^\lambda, x_i, R_i, \overline{\mathsf{msg}}^{<5})$

Theorem 5.1. *Assuming the existence of a public-coin perfectly correct trapdoor generation protocol, a perfectly correct* 3*-round special non-malleable commitment, a perfectly correct* 4*-round MPC protocol that is three-rewindable secure with abort against dishonest majority in the plain model, a perfectly correct* 2*-round public-coin WI proof, then* Π^{ID} *is a four-round MPC secure protocol with identifiable abort against dishonest majority in the plain model.*

The above construction Π^{ID} can be built based on trapdoor permutations (we refer to the full version for details on the instantiations of each of the building blocks).

Below, we describe the simulation. We denote the set that contains the indices of all the corrupted parties as $\mathcal{I}$. Before describing how our simulator $\mathcal{S}$ works, we define an algorithm $\mathcal{M}$ that we refer to as the *augmented machine*. The augmented machine internally runs the adversary $\mathcal{A}$ (we refer to this as the *left interface*), and acts as a proxy between $\mathcal{A}$ and its external interface (which we denote as the *right interface*) with respect to the messages of Π. At a high level, $\mathcal{M}$ filters the messages of Π that will be forwarded to the simulator of Π denoted by $\Pi.\mathcal{S}$, and forwards the replies received from $\Pi.\mathcal{S}$ to $\mathcal{A}$. The way in which $\mathcal{M}$ and $\Pi.\mathcal{S}$ interact with each other is regulated by our simulator $\mathcal{S}$, that internally runs (and has full control of) $\mathcal{M}$ and $\Pi.\mathcal{S}$.

The reason why we describe our simulator via the augmented machine $\mathcal{M}$ is to deal with the rewinds that the simulator of Π might do. We refer to Figure 5.2 and Figure 5.3 for the formal description of $\mathcal{M}$ and $\mathcal{S}$ respectively.

Figure 5.2: $\mathcal{M}(\rho_{\mathcal{A}}, \rho)$

Unless otherwise specified, in each round if a set of parties stops replying then all the honest parties stop and output `abort` together with the index of the party with the smallest index to indicate which is the aborting party.

Initialization Run $\mathcal{A}$ using the randomness $\rho_{\mathcal{A}}$ and use the randomness ρ to compute all the messages described below.

Round 1.

– Upon receiving the message $\mathsf{msg}_{1,i}$ from the right session where $i \in \overline{\mathcal{I}}$ do the following.
  1. Compute $\mathsf{td}_{1,i} \leftarrow \mathsf{TDGen}_1\,(R_{\mathsf{td},i})$.
  For every $j \neq i$:
  2. Compute $\mathsf{NMCom}_1^{i\to j} \leftarrow \mathsf{NMCom}_1(\tilde{r}^{i\to j}, R_{\mathsf{NMCom}}^{i\to j})$ where $\tilde{r}^{i\to j} \leftarrow \{0,1\}^{\lambda}$.
  3. Broadcast $\{\mathsf{NMCom}_1^{i\to j}\}_{j\in[n]\setminus\{i\}}, \mathsf{td}_{1,i}, \mathsf{msg}_{1,i}$.

– Upon receiving the first round from $\mathcal{A}$, for each $i \in \mathcal{I}$ forward $\mathsf{msg}_{1,i}$ to the right interface and continue as follows.

Round 2.

– Upon receiving the message $\mathsf{msg}_{2,i}$ from the right interface where $i \in \overline{\mathcal{I}}$ do the following:
  1. Compute the receiver message of TDGen: $\mathsf{td}_2^{i\to j} \leftarrow \mathsf{TDGen}_2(\mathsf{td}_{1,j})$.
  2. Compute the receiver message of the non-malleable commitment scheme $\mathsf{NMCom}_2^{j\to i} \leftarrow \mathsf{NMCom}_2(\mathsf{nmc}_1^{j\to i})$.

– Broadcast $\{\mathsf{NMCom}_1^{j\to i}, \mathsf{td}_2^{i\to j}\}_{j\in[n]\setminus\{i\}}, \mathsf{msg}_{2,i}$.

– Upon receiving the second round from $\mathcal{A}$, for each $i \in \mathcal{I}$ forward $\mathsf{msg}_{2,i}$ to the right interface and continue as follows.

Round 3.

– Upon receiving the message $\mathsf{msg}_{3,i}$ from the right session where $i \in \overline{\mathcal{I}}$ do the following:
  1. Compute the third round of TDGen: set $\mathsf{td}_{2,i} = \mathsf{td}_2^{1\to i}||\dots||\mathsf{td}_2^{n\to i}$ where $\mathsf{td}_2^{i\to i} = \perp$ and compute $\mathsf{td}_{3,i} \leftarrow \mathsf{TDGen}_3(\mathsf{td}_{2,i})$
     For every $j \neq i$:
     (a) Compute the third round of NMCom: $\mathsf{NMCom}_3^{i\to j} \leftarrow \mathsf{NMCom}_3(\mathsf{nmc}_2^{i\to j}, \tilde{r}^{i\to j}; R_{\mathsf{NMCom}}^{i\to j})$.
     (b) Compute the first round of RWI: $\mathsf{zap}_1^{j\to i} \leftarrow \{0,1\}^{\lambda}$.
  2. Broadcast $\{\mathsf{NMCom}_3^{i\to j}, \mathsf{zap}_1^{j\to i}\}_{j\in[n]\setminus\{i\}}, \mathsf{td}_{3,i}, \mathsf{msg}_{3,i}$

Upon receiving the third round from $\mathcal{A}$, for each $i \in \mathcal{I}$ do the following.

Check abort. On the behalf of the honest party P_i do the following: If $\exists j \neq i$ such that $\mathsf{TDValid}(\mathsf{td}_{1,j}, \mathsf{td}_{2,j}, \mathsf{td}_{3,j}) \neq 1$ then let j be the smallest of such indexes, send $(\mathtt{abort}, j)$ to the ideal functionality and output the view generated so far and stop. Else for each $i \in \overline{\mathcal{I}}$ do the following:

Check if the trapdoor has been already extracted. Send $\mathsf{get\_trap}$ to the right interface. If the reply received from the right interface is $0^{2\lambda}$ then go to **Rewinds**. Else, if the reply is $\{\mathsf{trap}_j\}_{j\in\mathcal{I}}$ such that for each $j \in \mathcal{I}$ $\mathsf{TDValid}(\mathsf{trap}_j, \mathsf{td}_j) = 1$ then send $\{\mathsf{msg}_{3,i}\}_{i\in\mathcal{I}}$ to the right interface, and upon receiving the message $\mathsf{msg}_{4,i}$ for all $i \in \overline{\mathcal{I}}$ from the right interface go to **Round 4**.

Rewinds.

1. Rewind $\mathcal{A}$ to the end of round 1 and freeze the main thread at this point. Then, create a set of T (to be determined later) rewinding threads, where on each thread, only rounds 2 and 3 of the protocol are executed using fresh randomness for each primitive.
2. For each look-ahead thread, define a thread to be GOOD with respect to P_i if for all malicious parties P_j
   - P_j does send its third round messages.
   - $\mathsf{TDValid}(\mathsf{td}_{1,j}, \mathsf{td}_{2,j}, \mathsf{td}_{3,j}) = 1$ where $\mathsf{td}_{2,j}$ is as computed in round 3.
3. The number of threads T created is such that at least 3 GOOD threads exist.

Trapdoor extraction.

1. For every corrupted party P_j , extract a trapdoor trap_j by running the trapdoor extractor TDExt using the transcript of the trapdoor generation protocol with P_j playing the role of the trapdoor generator from any 3 GOOD threads. Specifically, compute $\mathsf{trap}_j \leftarrow \mathsf{TDExt}(\mathsf{td}_1, \{\mathsf{td}_2^k, \mathsf{td}_3^k\}_{k=1}^3)$ where $(\mathsf{td}_1, \mathsf{td}_2^k, \mathsf{td}_3^k)$ denotes the transcript of the trapdoor generation protocol with P_j as the sender of the k-th GOOD thread.
2. Send $(\mathsf{set\_trap}, \{\mathsf{trap}_j\}_{j\in\mathcal{I}})$ to the right interface.
3. Go back to the pre-rewinds thread, send $\{\mathsf{msg}_{3,i}\}_{i\in\mathcal{I}}$ to the right interface. Upon receiving the message $\mathsf{msg}_{4,i}$ from the right interface for all $i \in \overline{\mathcal{I}}$ continue as follows

Round 4. For all $i \in \overline{\mathcal{I}}$

4. Set $r^{i\to j} \leftarrow \mathsf{trap}_j \oplus \tilde{r}^{i\to j}$
5. Compute RWI:
   Define $\mathtt{st} := \left(\overline{\mathsf{msg}}^{<5}, \left\{\mathsf{msg}_{\ell,i}\right\}_{\ell\in[4]}, \{\mathsf{nmc}_\ell^{i\to j}\}_{\ell\in[3]}, \mathsf{td}_{1,i}, r^{i\to j}\right)$ and $\mathtt{w} := \left(\perp, \perp, \tilde{r}^{i\to j}, R^{i\to j}_{\mathsf{NMCom}}\right)$ and compute $\mathsf{zap}_2^{i\to j} \leftarrow \mathcal{P}(\mathtt{st}, \mathtt{w}, \mathsf{zap}_1^{i\to j})$.

End of the simulation For all $i \in \overline{\mathcal{I}}$

1. If $\exists j \neq i$ and k, s.t. $\mathcal{V}(\mathtt{st}, \mathsf{zap}_2^{j\to k}, \mathsf{zap}_1^{j\to k}, \mathtt{st}) = 0$ where $\mathtt{st} := \left(\overline{\mathsf{msg}}^{<5}, \left\{\mathsf{msg}_{\ell,j}\right\}_{\ell\in[4]}, \{\mathsf{nmc}_\ell^{j\to k}\}_{\ell\in[3]}, \mathsf{td}_{1,k}, r^{j\to k}\right)$, let j be the smallest of such indexes such that this holds, then send $(\mathtt{abort}, j)$ to the ideal functionality and output the view generated so far.

Figure 5.3: $\mathcal{S}$

- Start $\mathcal{S}$ using randomness of appropriate length and initialize $\mathsf{trap} = 0^{2\lambda}$.
- Forward any query made by $\mathcal{S}$ directed to the adversarial interface to the right interface of $\mathcal{M}$.
- Upon receiving a query from $\mathcal{S}$ directed to the ideal functionality forward the query to it.

- Upon receiving $(\mathsf{set\_trap}, y)$ from $\mathcal{M}$ set $\mathsf{trap} \leftarrow y$.
- Upon receiving the command $\mathsf{get\_trap}$ from $\mathcal{M}$ send trap to the right interface of $\mathcal{M}$.
- Upon receiving the command $(\mathtt{abort}, j)$ from $\mathcal{M}$ forward it to the ideal functionality.
- Upon receiving the messages $\{\mathsf{msg}_{k,i}\}_{i \in \mathcal{I}}$ from the right interface of $\mathcal{M}$, rewind $\mathcal{S}$ up to the k-th round and forward $\{\mathsf{msg}_{k,i}\}_{i \in \mathcal{I}}$ to $\mathcal{S}$.[a]
- Whenever $\mathcal{S}$ stops, stop and output whatever $\mathcal{S}$ outputs.

[a] Note that $\mathcal{S}$ has full control of $\Pi.\mathcal{S}$, hence it can rewind $\Pi.\mathcal{S}$ at its will. In the security proof such rewinds need to be handled with some care as we will show in the formal proof.

The indistinguishability proof of Theorem 5.1 can be found in the full version.

6 Special B_{OT}-Rewindable Secure Oblivious Transfer

Now, we present a modified version of the compiler of Choudhuri et al. [10] that achieves our new notion of special B_{OT}-rewindable security and sender simulatability. The compiler makes use of the following tools:

- A delayed-input oblivious transfer protocol $\mathsf{OT}' = (\mathsf{OT}'_1, \mathsf{OT}'_2, \mathsf{OT}'_3, \mathsf{OT}'_4, \mathsf{OT}'_5)$ with the following properties: one-sided simulation security, and 2-extractability. In the full version we recall the oblivious transfer protocol of [34] and observe that it has the desired properties.
- A garbled circuit $(\mathsf{garble}, \mathsf{eval}, \mathsf{simGC})$ that is assumed to satisfy privacy, perfect correctness, authenticity of input labels and partial evaluation resiliency.

Figure 6.1: B_{OT}-rewindable Compiler OT

Private Inputs. The sender uses as its private input two lists (L_0, L_1), where each list consists of l bit strings of length λ, i.e. $L_b = \{y_{i,b}\}_{i \in [l]}$ and $y_{i,b} \in \{0,1\}^\lambda$. The receiver uses as its private input a vector $\vec{x}$ that consists of l bits, i.e. $\vec{x} = (x_1, \ldots, x_l)$.
Output. The receiver obtains the values $\{y_{i,x_i}\}_{i \in [l]}$.
Round 1. (Receiver)
1. Compute the first round message of all the OTs, i.e. for all $i \in [n], k \in [B_{\mathsf{OT}}], \mathsf{ot}_1^{i,k} \leftarrow \mathsf{OT}_1(1^\lambda, r^1_{i,k})$. We refer to index i as the outer index and k as the inner index.
2. Output $\{\mathsf{ot}_1^{i,k}\}_{i \in [n], k \in [B_{\mathsf{OT}}]}$ to the sender.

Round 2. (Sender)
1. Compute the second round message of all the OTs, i.e. for all $i \in [n], k \in [B_{\mathsf{OT}}], \mathsf{ot}_2^{i,k} \leftarrow \mathsf{OT}'_1(\mathsf{ot}_1^{i,k}; r^2_{i,k})$.
2. Output $\{\mathsf{ot}_2^{i,k}\}_{i \in [n], k \in [B_{\mathsf{OT}}]}$ to the receiver.

Round 3. (Receiver)

1. Encode the input $\vec{x}$ using n additive shares, i.e. sample $\vec{b}_j \leftarrow \{0,1\}^l$ for all $j \in [n-1]$ and compute $\vec{b}_n := \bigoplus_{j=1}^{n-1} \vec{b}_j$.
2. Select one of the OT' instances for all of the outer indexes i, i.e. sample $\sigma_i \leftarrow [B_{\mathsf{OT}}]$ for all $i \in [n]$.
3. Use the input $\vec{b}_i$ to compute $\mathsf{ot}_3^{i,\sigma_i} \leftarrow \mathsf{OT}'_3(\vec{b}_i, \{\mathsf{ot}_j^{i,\sigma_i}\}_{j\in[2]}; r_i^3)$. The other OTs are discontinued.
4. Output $\{\mathsf{ot}_3^{i,\sigma_i}\}_{i\in[n]}$ to the sender.

Round 4. (Sender)

1. Compute the garbled circuit $(\mathsf{GC}, \{K_{i,b}\}_{i\in[n],b\in\{0,1\}}) := \mathsf{garble}(\mathsf{C}_{\mathsf{OT}}[\{y_{i,0}\}_{i\in[l]}, \{y_{i,1}\}_{i\in[l]}])$, where the circuit $\mathsf{C}_{\mathsf{OT}}[\{y_{i,0}\}_{i\in[l]}, \{y_{i,1}\}_{i\in[l]}]$ on input $\vec{x}_1, \ldots, \vec{x}_l$ outputs $\{y_{i,x_i}\}_{i\in[l]}$ with $\vec{x} = (\vec{x}_1, \ldots, \vec{x}_l) := \bigoplus_{i=1}^{n} \vec{b}_i$.
2. For all $i \in [n]$, compute $\mathsf{ot}_4^{i,\sigma_i} \leftarrow \mathsf{OT}'_4(\{K_{i,b}\}_{b\in\{0,1\}}, \{\mathsf{ot}_j^{i,\sigma_i}\}_{j\in[3]}; r_i^4)$
3. Output $\{\mathsf{ot}_4^{i,\sigma_i}\}_{i\in[n]}$ and GC to the receiver.

Round 5. (Receiver)

1. Output $(r^1_{\sigma_i,i}, r_i^3)_{i\in[n]}$ to the sender.

Output Computation. (Receiver)

1. For all $i \in [n]$, compute $\widetilde{\mathsf{K}}_i := \mathsf{OT}'_5(\vec{b}_i, \{\mathsf{ot}_j^{i,\sigma_i}\}_{j\in[4]})$.
2. Output $\{y'_i\}_{i\in[l]} := \mathsf{eval}(\mathsf{GC}, \{\widetilde{\mathsf{K}}_i\}_{i\in[n]})$.

Theorem 6.1. *Assuming the existence of a delayed-input oblivious transfer protocol that satisfies one-sided simulation security, and 2-extractability, and a garbled circuit that satisfies privacy, perfect correctness, authenticity of input labels and partial evaluation resiliency, then the* OT *(described above) is an oblivious transfer protocol with special* B_{OT}*-rewindable and sender simulatability.*

The above construction can be built based on trapdoor permutations (we refer to the full version for details on the instantiations of each of the building blocks). In the remainder of this section we argue informally about sender simulatability, the proof can be found in the full version.

Sender Simulatability for parallel executions of OT. Our construction of B-rewindable secure MPC Π_{rmpc} in Section 4 uses the OT defined in this section as a building block. It is crucial for proving the security of Π_{rmpc} that it is possible to extract from all the malicious receivers at the same time. In more detail, the simulator $\mathcal{S}_{\mathsf{OT}}$ of OT proceeds to the extraction via rewinds (i.e. sending to the adversary a 2nd round of OT computed with new randomness), therefore the simulator of the OT should be able to extract the inputs of malicious receivers in the same rewinding thread for multiple OT instances that are executed in parallel. We refer to this property informally as *simultaneous extractability.* We therefore define $\mathcal{S}_{\mathsf{OT}}$ when multiple, say m^2, executions of the protocol OT of Figure 6.1 are executed in parallel, with a single execution corresponding to each pair of parties, where m denotes the number of parties.

Let us indicate with OT' the underlying OT protocol that is used in Figure 6.1. We discuss now the high-level overview of $\mathcal{S}_{\mathsf{OT}}$. Recall that in the rewind-secure OT construction of Figure 6.1, the OT receiver chooses a set of indices among the underlying OT' instances that it wishes to continue. However, the adversary acting on behalf of the OT receiver can choose to reveal a different set of indices across different rewinds of the simulator. Note that, to extract the receiver's input used in an instance of OT', the simulator requires two transcripts where the receiver chooses the same index. Therefore, in order to extract the input of malicious receivers across multiple OT instances in the same rewind thread, it is crucial that the indices for all the OT instances have appeared at least once previously in the rewind thread in which we are able to extract. Based on the above, the natural simulation strategy would be to continue rewinds until the above condition occurs i.e. until there occurs a rewinding thread such that the indices for all the OT instances have appeared at least once previously (we refer to this as a collision). It is important to notice that if in a rewinding thread a collision does not appear, then $\mathcal{S}_{\mathsf{OT}}$ obtains a transcript with a new index. Therefore, it is sufficient for $\mathcal{S}_{\mathsf{OT}}$ to rewind until it finds a collision. Unfortunately, simulation based on the above natural halting condition suffers from the following issue: if the simulator stops the first time it sees a transcript having the set of indices seen earlier, then the distribution of the transcripts output by the simulator is biased towards more frequently appearing indices that appeared in earlier rewinds. Therefore, the simulated view is not indistinguishable from the real view of the adversary. A similar issue was observed in the zero-knowledge protocol of [30], where the authors proposed a new halting condition of the simulator to maintain the indistinguishability of the views. We adopt their solution in the design of our simulation strategy. In more detail, for a better understanding, we elaborate on the issue of [30] and describe how their scenario is analogous to us. The authors of [30] consider N parallel instances of the following four-round ZK protocol: the verifier commits to a challenge in the 1st round and opens it in the 3rd round. In the 4th round the prover answers to the challenge. In particular, the verifier chooses t among the N instances for which it opens the challenge. The prover, in round 4, finishes the ZK protocol for the revealed challenges. The simulator S on behalf of the honest prover should be able to cheat w.r.t. all the indices (of the instances) opened by the verifier. In order to do that S rewinds the malicious verifier V^*, and succeeds if each of the t indices opened by V^* has appeared in at least one of the earlier rewinds (a collision occurs). Note that V^* can choose different indices during different rewinds. However, when S fails to cheat it learns at least one new index that did not occur in any of the previous rewinds. In [30] it is pointed out that if S stops as soon as it can simulate successfully, then the simulated view could be distinguishable from the real world view of V^* due to the same reasons explained above (in the context of OT). It is easy to see that $\mathcal{S}_{\mathsf{OT}}$ could collect the indices in the same way as S. It remains to argue that the strategies with which the malicious receiver can open the OT' indices are a subset of the one that V^* can perform. Consider an execution of OT where the malicious receiver opens n indices $k_1, \dots, k_n$ as opposed to t indices

$i_1, \ldots, i_t$ given by V^*. It easy to see that N of [30] corresponds to nB in our case. For simplicity, let us assume that the underlying OT' instances are labeled $\{1, \ldots, nB\}$, then each k_j has a value only in $\{(j-1) \cdot B + 1, \ldots, j \cdot B\}$, with $j \in [n]$. Therefore, this corresponds to a strategy of V^* where for each i_j, V^* chooses value only in $\{j - 1 \cdot (\frac{N}{t}) + 1, \ldots, j \cdot \frac{N}{t}\}$, with $j \in [t]$.

Acknowledgements. We are grateful to Maciej Obremski, our tall lighthouse who illuminated our path toward proving the complexity of our simulators.

References

1. Ananth, P., Choudhuri, A.R., Goel, A., Jain, A.: Two round information-theoretic MPC with malicious security. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part II. LNCS, vol. 11477, pp. 532–561. Springer, Heidelberg (May 2019), DOI: 10.1007/978-3-030-17656-3_19
2. Ananth, P., Choudhuri, A.R., Jain, A.: A new approach to round-optimal secure multiparty computation. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 468–499. Springer, Heidelberg (Aug 2017), DOI: 10.1007/978-3-319-63688-7_16
3. Aumann, Y., Lindell, Y.: Security against covert adversaries: Efficient protocols for realistic adversaries. Journal of Cryptology **23**(2), 281–343 (Apr 2010), DOI: 10.1007/s00145-009-9040-7
4. Badrinarayanan, S., Goyal, V., Jain, A., Kalai, Y.T., Khurana, D., Sahai, A.: Promise zero knowledge and its applications to round optimal MPC. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 459–487. Springer, Heidelberg (Aug 2018), DOI: 10.1007/978-3-319-96881-0_16
5. Baum, C., Orsini, E., Scholl, P., Soria-Vazquez, E.: Efficient constant-round MPC with identifiable abort and public verifiability. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 562–592. Springer, Heidelberg (Aug 2020), DOI: 10.1007/978-3-030-56880-1_20
6. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: 22nd ACM STOC. pp. 503–513. ACM Press (May 1990), DOI: 10.1145/100216.100287
7. Brakerski, Z., Halevi, S., Polychroniadou, A.: Four round secure computation without setup. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 645–677. Springer, Heidelberg (Nov 2017), DOI: 10.1007/978-3-319-70500-2_22
8. Brandt, N.: Tight setup bounds for identifiable abort. Cryptology ePrint Archive, Report 2021/684 (2021), `https://eprint.iacr.org/2021/684`
9. Brandt, N.P., Maier, S., Müller, T., Müller-Quade, J.: Constructing secure multiparty computation with identifiable abort. Cryptology ePrint Archive, Report 2020/153 (2020), `https://eprint.iacr.org/2020/153`
10. Choudhuri, A.R., Ciampi, M., Goyal, V., Jain, A., Ostrovsky, R.: Round optimal secure multiparty computation from minimal assumptions. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 291–319. Springer, Heidelberg (Nov 2020), DOI: 10.1007/978-3-030-64378-2_11
11. Ciampi, M., Ostrovsky, R., Siniscalchi, L., Visconti, I.: Delayed-input nonmalleable zero knowledge and multi-party coin tossing in four rounds. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 711–742. Springer, Heidelberg (Nov 2017), DOI: 10.1007/978-3-319-70500-2_24

12. Ciampi, M., Ostrovsky, R., Siniscalchi, L., Visconti, I.: Round-optimal secure two-party computation from trapdoor permutations. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 678–710. Springer, Heidelberg (Nov 2017), DOI: 10.1007/978-3-319-70500-2_23
13. Cleve, R.: Limits on the security of coin flips when half the processors are faulty (extended abstract). In: 18th ACM STOC. pp. 364–369. ACM Press (May 1986), DOI: 10.1145/12130.12168
14. Cohen, R., Garay, J.A., Zikas, V.: Broadcast-optimal two-round MPC. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 828–858. Springer, Heidelberg (May 2020), DOI: 10.1007/978-3-030-45724-2_28
15. Cohen, R., Lindell, Y.: Fairness versus guaranteed output delivery in secure multiparty computation. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part II. LNCS, vol. 8874, pp. 466–485. Springer, Heidelberg (Dec 2014), DOI: 10.1007/978-3-662-45608-8_25
16. Cunningham, R.K., Fuller, B., Yakoubov, S.: Catching MPC cheaters: Identification and openability. In: Shikata, J. (ed.) ICITS 17. LNCS, vol. 10681, pp. 110–134. Springer, Heidelberg (Nov / Dec 2017)
17. Damgård, I., Keller, M., Larraia, E., Pastro, V., Scholl, P., Smart, N.P.: Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In: Crampton, J., Jajodia, S., Mayes, K. (eds.) ESORICS 2013. LNCS, vol. 8134, pp. 1–18. Springer, Heidelberg (Sep 2013), DOI: 10.1007/978-3-642-40203-6_1
18. Damgård, I., Magri, B., Ravi, D., Siniscalchi, L., Yakoubov, S.: Broadcast-optimal two round MPC with an honest majority. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part II. LNCS, vol. 12826, pp. 155–184. Springer, Heidelberg, Virtual Event (Aug 2021), DOI: 10.1007/978-3-030-84245-1_6
19. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 643–662. Springer, Heidelberg (Aug 2012), DOI: 10.1007/978-3-642-32009-5_38
20. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). In: 23rd ACM STOC. pp. 542–552. ACM Press (May 1991), DOI: 10.1145/103418.103474
21. Dwork, C., Naor, M.: Zaps and their applications. In: 41st FOCS. pp. 283–293. IEEE Computer Society Press (Nov 2000), DOI: 10.1109/SFCS.2000.892117
22. Garg, S., Mukherjee, P., Pandey, O., Polychroniadou, A.: The exact round complexity of secure computation. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 448–476. Springer, Heidelberg (May 2016), DOI: 10.1007/978-3-662-49896-5_16
23. Garg, S., Sahai, A.: Adaptively secure multi-party computation with dishonest majority. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 105–123. Springer, Heidelberg (Aug 2012), DOI: 10.1007/978-3-642-32009-5_8
24. Goldreich, O., Krawczyk, H.: On the composition of zero-knowledge proof systems. SIAM J. Comput. **25**(1), 169–192 (1996)
25. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A. (ed.) 19th ACM STOC. pp. 218–229. ACM Press (May 1987), DOI: 10.1145/28395.28420
26. Goyal, V.: Constant round non-malleable protocols using one way functions. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd ACM STOC. pp. 695–704. ACM Press (Jun 2011), DOI: 10.1145/1993636.1993729

27. Goyal, V., Pandey, O., Richelson, S.: Textbook non-malleable commitments. In: Wichs, D., Mansour, Y. (eds.) 48th ACM STOC. pp. 1128–1141. ACM Press (Jun 2016), DOI: 10.1145/2897518.2897657
28. Goyal, V., Richelson, S., Rosen, A., Vald, M.: An algebraic approach to non-malleability. In: 55th FOCS. pp. 41–50. IEEE Computer Society Press (Oct 2014), DOI: 10.1109/FOCS.2014.13
29. Halevi, S., Hazay, C., Polychroniadou, A., Venkitasubramaniam, M.: Round-optimal secure multi-party computation. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 488–520. Springer, Heidelberg (Aug 2018), DOI: 10.1007/978-3-319-96881-0_17
30. Hazay, C., Venkitasubramaniam, M.: Round-optimal fully black-box zero-knowledge arguments from one-way permutations. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part I. LNCS, vol. 11239, pp. 263–285. Springer, Heidelberg (Nov 2018), DOI: 10.1007/978-3-030-03807-6_10
31. Ishai, Y., Ostrovsky, R., Seyalioglu, H.: Identifying cheaters without an honest majority. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 21–38. Springer, Heidelberg (Mar 2012), DOI: 10.1007/978-3-642-28914-9_2
32. Ishai, Y., Ostrovsky, R., Zikas, V.: Secure multi-party computation with identifiable abort. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 369–386. Springer, Heidelberg (Aug 2014), DOI: 10.1007/978-3-662-44381-1_21
33. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer - efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (Aug 2008), DOI: 10.1007/978-3-540-85174-5_32
34. Katz, J., Ostrovsky, R.: Round-optimal secure two-party computation. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 335–354. Springer, Heidelberg (Aug 2004), DOI: 10.1007/978-3-540-28628-8_21
35. Katz, J., Ostrovsky, R., Smith, A.: Round efficiency of multi-party computation with a dishonest majority. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 578–595. Springer, Heidelberg (May 2003), DOI: 10.1007/3-540-39200-9_36
36. Khurana, D.: Round optimal concurrent non-malleability from polynomial hardness. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part II. LNCS, vol. 10678, pp. 139–171. Springer, Heidelberg (Nov 2017), DOI: 10.1007/978-3-319-70503-3_5
37. Kilian, J.: Founding cryptography on oblivious transfer. In: 20th ACM STOC. pp. 20–31. ACM Press (May 1988), DOI: 10.1145/62212.62215
38. Lin, H., Pass, R., Venkitasubramaniam, M.: Concurrent non-malleable commitments from any one-way function. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 571–588. Springer, Heidelberg (Mar 2008), DOI: 10.1007/978-3-540-78524-8_31
39. Pass, R.: Bounded-concurrent secure multi-party computation with a dishonest majority. In: Babai, L. (ed.) 36th ACM STOC. pp. 232–241. ACM Press (Jun 2004), DOI: 10.1145/1007352.1007393
40. Pass, R., Rosen, A.: Concurrent non-malleable commitments. In: 46th FOCS. pp. 563–572. IEEE Computer Society Press (Oct 2005), DOI: 10.1109/SFCS.2005.27
41. Pass, R., Wee, H.: Constant-round non-malleable commitments from sub-exponential one-way functions. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 638–655. Springer, Heidelberg (May / Jun 2010), DOI: 10.1007/978-3-642-13190-5_32

42. Scholl, P., Simkin, M., Siniscalchi, L.: Multiparty computation with covert security and public verifiability. Cryptology ePrint Archive, Report 2021/366 (2021), https://eprint.iacr.org/2021/366
43. Simkin, M., Siniscalchi, L., , Yakoubov, S.: On sufficient oracles for secure computation with identifiable abort. Cryptology ePrint Archive, Report 2021/151 (2021), https://eprint.iacr.org/2021/151
44. Spini, G., Fehr, S.: Cheater detection in SPDZ multiparty computation. In: Nascimento, A.C.A., Barreto, P. (eds.) ICITS 16. LNCS, vol. 10015, pp. 151–176. Springer, Heidelberg (Aug 2016), DOI: 10.1007/978-3-319-49175-2_8
45. Wee, H.: Black-box, round-efficient secure computation via non-malleability amplification. In: 51st FOCS. pp. 531–540. IEEE Computer Society Press (Oct 2010), DOI: 10.1109/FOCS.2010.87
46. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS. pp. 162–167. IEEE Computer Society Press (Oct 1986), DOI: 10.1109/SFCS.1986.25