Round Optimal Secure Multiparty Computation from Minimal Assumptions

Arka Rai Choudhuri[1](✉), Michele Ciampi[2], Vipul Goyal[3,4], Abhishek Jain[1], and Rafail Ostrovsky[5]

[1] Johns Hopkins University, Baltimore, USA
{achoud,abhishek}@cs.jhu.edu
[2] The University of Edinburgh, Edinburgh, UK
mciampi@ed.ac.uk
[3] Carnegie Mellon University, Pittsburgh, USA
goyal@cs.cmu.edu
[4] NTT Research, East Palo Alto, USA
[5] University of California, Los Angeles, USA
rafail@cs.ucla.edu

Abstract. We construct a four round secure multip arty computation (MPC) protocol in the plain model that achieves security against any dishonest majority. The security of our protocol relies only on the existence of four round oblivious transfer. This culminates the long line of research on constructing round-efficient MPC from minimal assumptions (at least w.r.t. black-box simulation).

1 Introduction

The ability to securely compute on private datasets of individuals has wide applications of tremendous benefits to society. Secure multiparty computation (MPC) [18,37] provides a solution to the problem of computing on private data by allowing a group of parties to jointly evaluate any function over their private inputs in such a manner that no one learns anything beyond the output of the function.

Since its introduction nearly three decades ago, MPC has been extensively studied along two fundamental lines: necessary *assumptions* [18,26,30], and *round complexity* [2,4,6,8,9,13,18,21,28,29,32,33,36].[1] Even for the case of malicious adversaries who may corrupt any number of parties, both of these topics, individually, are by now pretty well understood:

- It is well known that oblivious transfer (OT) is both necessary and sufficient [26,30] for MPC.
- A recent sequence of works have established that *four rounds* are both necessary [13] and sufficient [2,3,6,25] for MPC (with respect to black-box simulation). However, the assumptions required by these works are far from optimal, ranging from subexponential hardness assumptions [2,6] to polynomial hardness of specific forms of encryption schemes [25] or specific number-theoretic assumptions [3].

[1] A detailed discussion on related works can be found in the full version.

R. Pass and K. Pietrzak (Eds.): TCC 2020, LNCS 12551, pp. 291–319, 2020.
https://doi.org/10.1007/978-3-030-64378-2_11

In this work, we consider the well studied goal of building round-efficient MPC while minimizing the underlying cryptographic assumptions. Namely:

Can we construct round optimal MPC from minimal assumptions?

Precisely, we ask whether it is possible to construct four round MPC from four round OT. This was explicitly left as an open problem in the elegant work of Benhamouda and Lin [5] who constructed k-round MPC from k-round OT for $k \geq 5$.

1.1 Our Results

In this work, we resolve the above question in the affirmative. Namely, we construct four round malicious-secure MPC based only on four round (malicious-secure) OT. Our protocol admits black-box simulation and achieves security against malicious adversaries in the dishonest majority setting.

Theorem 1 (Informal). *Assuming the existence of four round OT, there exists a four round MPC protocol for any efficiently computable functionality in the plain model.*

This settles the long line of research on constructing round efficient MPC from minimal cryptographic assumptions.

Our Approach. To obtain our result, we take a conceptually different approach from the works of [2,3,6,25] for enforcing honest behavior on (possibly malicious) protocol participants. Unlike these works, we do not require the parties to give an *explicit* proof of honest behavior within the first three rounds of the protocol. Instead, we devise a *multiparty conditional disclosure of secrets* mechanism that ensures that the final round messages of the honest parties become "opaque" if even a single participant behaved maliciously. A key property of this mechanism is that it allows for each party to obtain a *public* witness that attests to honest behavior of all the parties, without compromising the security of any party. We refer the reader to Sect. 2 for details.

On the Minimal Assumptions. We study MPC in the standard broadcast communication model, where in each round, every party broadcasts a message to the other parties. In this model, k-round MPC implies k-round *bidirectional* OT, where each round consists of messages from both the OT sender and the receiver. However, it is not immediately clear whether it also implies k-round OT in the standard, *alternating-message* model for two-party protocols where each round consists of a message from only one of the two parties. As such, the minimal assumption for k-round MPC is, in fact, k-round bidirectional OT (as opposed to alternating-message OT).

Towards establishing the optimality of Theorem 1, we observe that k-round bidirectional OT implies k-round alternating-message OT.

Theorem 2. *k-round bidirectional OT implies k-round alternating-message OT.*

Our transformation is unconditional and generalizes a message rescheduling strategy previously considered by Garg et al. [13] for the specific case of three round coin-tossing protocols. In fact, this transformation is even more general and applies to any two-party functionality, with the restriction that only one party learns the output in the alternating- essage protocol.

An important corollary of Theorem 2 is that it establishes the missing piece from the result of Benhamouda and Lin [5] who constructed k-round MPC from any k-round alternating-message OT for $k \geq 5$. Their result, put together with our main result in Theorem 1 provides a *full resolution* of the fundamental question of basing round efficient MPC on minimal assumptions.

In the sequel, for simplicity of exposition, we refer to alternating-message OT as simply OT.

2 Technical Overview

Before we dive into the technical contributions of our work, for the uninitiated reader, we provide a brief summary of the key challenges that arise in the design of a four round MPC protocol and the high-level strategies adopted in prior works for addressing them. We group these challenges into three broad categories, and will follow the same structure in the remainder of the section.

Enforcing Honest Behavior. A natural idea, adopted in prior works, is to start with a protocol that achieves security against semi-malicious[2] adversaries and compile it using zero-knowledge (ZK) proofs [20] à la GMW compiler [19] to achieve security against malicious adversaries. This is not easy, however, since we are constrained by the number of rounds. As observed in prior works, when the underlying protocol is *delayed* semi-malicious[3] [2,5], we can forego establishing honest behavior in the first two rounds. In particular, it suffices to establish honest behavior in the third and fourth rounds. The main challenge that still persists, however, is that ZK proofs – the standard tool for enforcing honest behavior – are impossible in three rounds w.r.t. black-box simulation [17]. Thus, an alternative mechanism is required for establishing honest behavior in the third round.

Need for Rewind Security. Due to the constraint on the number of rounds, all prior works utilize design templates where multiple sub-protocols are executed in *parallel*. This creates a challenge when devising a black-box simulation strategy that works by rewinding the adversary. In particular, if the simulator rewinds the adversary (say) during second and third round of the protocol, e.g., to extract its input, we can no longer rely on stand-alone security of sub-protocols used in those rounds. This motivates the

[2] Roughly speaking, such adversaries behave like semi-honest adversaries, except that they may choose arbitrary random tapes.

[3] Roughly speaking, a delayed semi-malicious adversary is similar to semi-malicious adversary, except that in the second last round of a k-round protocol, it is required to output (on a special tape) a witness (namely, its input and randomness) that establishes its honest behavior in all the rounds so far.

use of sub-protocols that retain their security even in the presence of some number of rewinds. Indeed, much work is done in all prior works to address this challenge.

Non-malleability. For similar reasons as above, we can no longer rely on standard soundness guarantee of ZK proofs (which only hold in the stand-alone setting). All prior works address this challenge via a careful use of some non-malleable primitive such as non-malleable commitments [10] in order to "bootstrap non-malleability" in the entire protocol. This leads to an involved security analysis.

Our primary technical contribution is in addressing the first two issues. We largely follow the template of prior works in addressing non-malleability challenges. As such, In the remainder of this technical overview, we focus on the first two issues, and defer discussion on non-malleability to the full version.

Organization. We describe our key ideas for tackling the first and second issues in Sects. 2.1 and 2.2, respectively. We conclude by providing a summary of our protocol in Sect. 2.3.

Full Version. Due to space constraints, preliminaries, details of the proofs, and complexity calculations have been omitted from this manuscript, and can be found in the full version of the paper [7].

2.1 Enforcing Honest Behavior

In any four round protocol, a rushing adversary may always choose to *abort* after receiving the messages of honest parties in the last round. At this point, the adversary has already received enough information to obtain the output of the function being computed. This suggests that we must enforce "honest behavior" on the protocol participants *within the first three rounds* in order to achieve security against malicious adversaries. Indeed, without any such safeguard, a malicious adversary may be able learn the inputs of the honest parties, e.g., by acting maliciously so as to change the functionality being computed to the identity function.

Since three-round ZK proofs with black-box simulation are known to be impossible, all recent works on four round MPC devise non-trivial strategies that only utilize weaker notions of ZK (that are achievable in three or less rounds) to enforce honest behavior within the first three rounds of the MPC protocol. However, all of these approaches end up relying on assumptions that are far from optimal: [2] and [6] use super-polynomial-time hardness assumptions, [25] use Zaps [11] and affine-homomorphic encryption schemes, and [3] use a new notion of promise ZK together with three round strong WI [27], both of which require specific number-theoretic assumptions.

A Deferred Verification Approach. We use a different approach to address the above challenge. We do not require the parties to give an explicit proof of honest behavior within the first three rounds. Of course, this immediately opens up the possibility for an adversary to cheat in the first three rounds in such a manner that by observing the messages of the honest parties in the fourth round, it can completely break privacy. To prevent such an attack, we require the parties to "encrypt" their last round message in such a manner that it can only be decrypted by using a "witness" that establishes honest behavior in the first three rounds. In other words, the verification check for honest behavior is deferred to the fourth round.

In the literature, the above idea is referred to as conditional disclosure of secrets (CDS) [1]. Typically, however, CDS is defined and constructed as a two-party protocol involving a single encryptor – who encrypts a secret message w.r.t. some statement – and a single decryptor who presumably holds a witness that allows for decryption.[4]

This does not suffice in the multiparty setting due to the following challenges:

- The multiparty setting involves multiple decryptors as opposed to a single decryptor. A naive way to address this would be to simply run multiple executions of two-party CDS in parallel, each involving a different decryptor, such that the i^{th} execution allows party i to decrypt by using a witness that establishes its own honest behavior earlier in the protocol. However, consider the case where the adversary corrupts at least two parties. In the above implementation, a corrupted party who behaved honestly during the first three rounds would be able to decrypt the honest party message in the last round even if another corrupted party behaved maliciously. This would clearly violate security. As such, we need a mechanism to *jointly* certify honest behavior of *all* the parties (as opposed to a single party).
- In the two-party setting, the input and randomness of the decryptor constitutes a natural witness for attesting its honest behavior. In the multiparty setting, however, it is not clear how an individual decryptor can obtain such a witness that establishes honest behavior of all the parties without trivially violating privacy of other parties.

We address these challenges by implementing a *multiparty conditional disclosure of secrets* (MCDS) mechanism. Informally speaking, an MCDS scheme can be viewed as a tuple of (possibly interactive) algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$: (a) Gen takes as input an instance and witness pair (x, w) and outputs a "public" witness π. (b) Enc takes as input n statements $(x_1, \ldots, x_n)$ and a message m and outputs an encryption c of m. (c) Dec takes as input a ciphertext c and tuples $(x_1, \pi_1), \ldots, (x_n, \pi_n)$ and outputs m or $\perp$. We require the following properties:

- **Correctness:** If all the instances $(x_1, \ldots, x_n)$ are true, then dec outputs m.
- **Message Privacy:** If at least one instance is false, then c is semantically secure.
- **Witness Privacy:** There exists a simulator algorithm that can simulate the output π of Gen without using the private witness w.

The security properties of MCDS allow us to overcome the aforementioned challenges. In particular, the witness privacy guarantee allows the parties to publicly release the witnesses $(\pi_1, \ldots, \pi_n)$ while maintaining privacy of their inputs and randomness.

In order to construct MCDS with witness privacy guarantee, we look towards ZK proof systems. As a first attempt, we could implement public witnesses via a delayed-input[5] four round ZK proof system. Specifically, each party i is required to give a ZK

[4] There are some exceptions; we refer the reader to the full version for discussion on other models.

[5] A proof system is said to be delayed input if the instance is only required for computing the last round of the proof.

proof for x_i such that the last round of the proof constitutes a public witness π_i. Further, a simple, non-interactive method to implement the encryption and the decryption mechanism is witness encryption [12]. However, presently witness encryption is only known from non-standard assumptions (let alone OT).

To achieve our result from minimal assumptions, we instead use garbled circuits [37] and four round OT to implement MCDS. Namely, each party i garbles a circuit that contains hardwired the entire transcript of the first three rounds of the underlying MPC, as well the fourth round message of the MPC of party i. Upon receiving as input a witness $\pi_1, \ldots, \pi_n$, where π_j is a witness for honest behavior of party j, the garbled circuit outputs the fourth round message. Each party j can encode its witness π_j in the OT receiver messages, where the corresponding sender inputs will be the wire labels of the garbled circuit. Party j then release its private randomness used inside OT in the fourth round so that any other party j' can use it to compute the output of the OT, thereby learning the necessary wire labels for evaluating the garbled circuit sent by party i. For security, it is imperative the witness π_j remains hidden until the randomness is revealed in the fourth round.

A problem with the above strategy is that in a four round OT, the receiver's input must be fixed by the third round. This means that we can no longer use four round ZK proofs, and instead must use *three round* proofs to create public witnesses of honest behavior. But which three round proofs must we use? Towards this, we look to the weaker notion of *promise* ZK introduced by [3]. Roughly, promise ZK relaxes the standard notion of ZK by guaranteeing security only against malicious verifiers who do *not* abort. Importantly, unlike standard ZK, distributional[6] promise ZK can be achieved in only three rounds with black-box simulation in the bidirectional message model. This raises two questions – is promise ZK sufficient for our purposes, and what assumptions are required for three round promise ZK?

Promise ZK Under the Hood. Let us start with the first question. An immediate challenge with using promise ZK is that it provides no security in the case where the verifier always aborts. In application to four round MPC, this corresponds to the case where the (rushing) adversary always aborts in the third round. Since the partial transcript at the end of third round (necessarily) contains inputs of honest parties, we still need to argue security in this case. The work of [3] addressed this problem by using a "hybrid" ZK protocol that achieves the promise ZK property when the adversary is non-aborting, and strong witness-indistinguishability (WI) property against aborting adversaries. The idea is that by relying on strong WI property (only in the case where adversary aborts in the third round), we can switch from using real inputs of honest parties to input 0. However, three round strong WI is only known based on specific number-theoretic assumptions [27].

To minimize our use of assumptions, we do *not* use strong WI, and instead devise a hybrid argument strategy – similar to that achieved via strong WI – by using promise ZK *under the hood*. Recall that since we use the third round prover message of promise ZK as a witness for conditional decryption, it is not given in the clear, but is instead "encrypted" inside the OT receiver messages in the third round. This has the positive effect of shielding promise ZK from the case where the adversary always aborts in the

[6] That is, where the instances are sampled from a public distribution.

third round.[7] In particular, we can use the following strategy for arguing security against aborting adversaries: we first switch from using promise ZK third round prover message to simply using 0's as the OT receiver's inputs. Now, we can replace the honest parties' inputs with 0 inputs by relying on the security of the sub-protocols used within the first three rounds. Next, we can switch back to using honestly computed promise ZK third round prover message as the OT receiver's inputs.

Let us now consider the second question, namely, the assumptions required for three round promise ZK. The work of [3] used specific number-theoretic assumptions to construct three round (distributional) promise ZK. However, we only wish to rely on the use of four round OT. Towards this, we note that the main ingredient in the construction of promise ZK by [3] that necessitated the use of number-theoretic assumptions is a three round WI proof system that achieves "bounded-rewind-security." Roughly, this means that the WI property holds even against verifiers who can rewind the prover an a priori bounded number of times.

Towards minimizing assumptions, we note that a very recent work of [23] provides a construction of such a WI based only on non-interactive commitments. By using their result, we can obtain three round promise ZK based on non-interactive commitments, which in turn can be obtained from four round OT using the recent observation of Lombardi and Schaeffer [31].

2.2 Rewinding Related Challenges

While the above ideas form the basis of our approach, we run into several obstacles during implementation due to rewinding-related issues that we mentioned earlier. In order to explain these challenges and our solution ideas, we first describe a high-level template of our four round MPC protocol based on the ideas discussed so far. To narrow the focus of the discussion on the challenges unique to the present work, we ignore some details for now and discuss them later.

An Initial Protocol Template. We devise a compiler from four round delayed semi-malicious MPC protocols of a special form to a four round malicious-secure MPC protocol. Specifically, we use a four round delayed semi-malicious protocol Π obtained by plugging in a four-round malicious-secure (which implies delayed semi-malicious security) OT in the k-round semi-malicious MPC protocol of [5,14] based on k-round semi-malicious OT. An important property of this protocol that we rely upon is that it consists only of OT messages in the first $k-2$ rounds. Further, we also rely upon the random self-reducibility of OT, which implies that the first two rounds do not depend on the OT receiver's input, and the first three rounds do not depend on the sender's input.[8]

To achieve malicious security, similar to prior works, our compiler uses several building blocks (see Sect. 2.3 for a detailed discussion). One prominent building block is a three-round extractable commitment scheme that is executed in *parallel* with the first three rounds of the delayed semi-malicious MPC. The extractable commitment

[7] Note that if the protocol does progress to the fourth round, then we do not need to shield promise ZK anymore.

[8] We note that this property was also used by [5] in their construction of k-round malicious-secure MPC.

scheme is used by the parties to commit to their inputs and randomness. This allows the simulator for our protocol to extract the adversary's inputs and randomness *by rewinding the second and third rounds*, and then use them to simulate the delayed semi-malicious MPC.

Bounded-Rewind-Secure OT. The above template poses an immediate challenge in proving security of the protocol. Since the simulator rewinds the second and third rounds in order to extract the adversary's inputs, this means that the second and third round messages of the delayed semi-malicious MPC also get rewound. For this reason, we cannot simply rely upon delayed semi-malicious security of the MPC. Instead, we need the MPC protocol to remain secure *even when it is being rewound*. More specifically, since we are using an MPC protocol where the first two rounds only consist of OT messages, we need a *four round rewind-secure OT protocol*. Since the third round of a four round OT only contains a message from the OT receiver, we need the following form of rewind security property: an adversarial sender cannot determine the input bit used by the receiver even if it can rewind the receiver during the second and third round.

Clearly, an OT protocol with black-box simulation cannot be secure against an arbitrary number of rewinds. In particular, the best we can hope for is security against an a priori *bounded* number of rewinds. Following observations from [3], we note that bounded-rewind security of OT is, in fact, *sufficient* for our purposes. Roughly, the main idea is that the rewind-security of OT is invoked to argue indistinguishability of two consecutive hybrids inside our security proof. In order to establish indistinguishability by contradiction, it suffices to build an adversary that breaks OT security with some non-negligible probability (as opposed to overwhelming probability). This, in turn means that the reduction only needs to extract the adversary's input required for generating its view with non-negligible probability. By using a specific extractable commitment scheme, we can ensure that the number of rewinds necessary for this task are a priori bounded.

Standard OT protocols, however, do not guarantee any form of bounded-rewind security. Towards this, we provide a generic construction of a four round bounded-rewind secure OT starting from any four round OT, which may be of independent interest. Our transformation is in fact more general and works for any $k \geq 4$ round OT, when rewinding is restricted to rounds $k-2$ and $k-1$. For simplicity, we describe our ideas for the case where we need security against *one* rewind; our transformation easily extends to handle more rewinds.

A natural idea to achieve one-rewind security for receivers, previously considered in [5], is the following: run two copies of an OT protocol in parallel for the first $k-2$ rounds. In round $k-1$, the receiver randomly chooses one of the two copies and only continues that OT execution, while the sender continues both the OT executions. In the last round, the parties only complete the OT execution that was selected by the receiver in round $k-1$. Now, suppose that an adversarial sender rewinds the receiver in rounds $k-2$ and $k-1$. Then, if the receiver selects different OT copies on the "main" execution thread and the "rewound" execution thread, we can easily reduce one-rewind security of this protocol to stand-alone security of the underlying OT.

The above idea suffers from a subtle issue. Note that the above strategy for dealing with rewinds is inherently *biased*, namely, the choice made by the receiver on the rewound thread is *not* random, and is instead correlated with its choice on the main thread. If we use this protocol in the design of our MPC protocol, it leads to the following issue during simulation: consider an adversary who chooses a random z and then always aborts if the receiver selects the z-th OT copy. Clearly, this adversary only aborts with probability $1/2$ in an honest execution. Now, consider the high-level simulation strategy for our MPC protocol discussed earlier, where the simulator rewinds the second and third rounds to extract the adversary's inputs. In order to ensure rewind security of the OT, this simulator, with overall probability $1/2$, will select the z-th OT copy on *all* the rewound execution threads. However, in this case, the simulator will always *fail* in extracting the adversary's inputs no matter how many times it rewinds.

We address the above problem via a secret-sharing approach to eliminate the bias. Instead of simply running two copies of OT, we run $\ell \cdot n$ copies in parallel during the first $k-2$ rounds. These $\ell \cdot n$ copies can be divided into n tuples, each consisting of ℓ copies. In round $k-1$, the receiver selects a single copy from each of the n tuples at random. It then uses n-out-of-n secret sharing to divide its input bit b into n shares $b_1, \ldots, b_n$, and then uses share b_i in the OT copy selected from the i-th tuple. In the last round, sender now additionally sends a garbled circuit (GC) that contains its input (x_0, x_1) hardwired. The GC takes as input all the bits $b_1, \ldots, b_n$, reconstructs b and then outputs x_b. The sender uses the labels of the GC as its inputs in the OT executions. Intuitively, by setting ℓ appropriately, we can ensure that for at least one tuple i, the OT copies randomly selected by the receiver on the main thread and the rewound threads are different, which ensures that b_i (and thereby, b) remains hidden. We refer the reader to the technical section for more details.

Proofs Of Proofs. We now describe another challenge in implementing our template of four round MPC. As discussed earlier, we use a three round extractable commitment scheme to enable extraction of the adversary's inputs and randomness. For reasons similar to those as for the case of OT, we actually use an extractable commitment scheme that achieves bounded-rewind security. Specifically, we use a simplified variant of the three-round commitment scheme constructed by [3].[9]

A specific property of this commitment scheme is that in order to achieve rewind security, it is designed such that the third round message of the committer is not "verifiable." This means that the committer may be able to send a malformed message without being detected by the receiver. For this reason, we require each party to prove the "well-formedness" of its commitment via promise ZK. This, however, poses the following challenge during simulation: since the third round prover message of promise ZK is encrypted inside OT receiver message, the simulator doesn't know whether the adversary's commitment is well-formed or not. In particular, if the adversary's commitment is not well-formed, the simulator may end up running *forever*, in its attempt to extract the adversary's input via rewinding.

[9] The commitment scheme of [3] also achieves some security properties, in addition to bounded rewind security, that are not required by our compiler. Hence, we use a simplified variant of their scheme.

One natural idea to deal with this issue is to first extract adversary's promise ZK message from the OT executions via rewinding, and then decide whether or not to attempt extracting the adversary's input. However, since we are using an *arbitrary* (malicious-secure) OT, we do *not* know in advance the number of rewinds required for extracting the receiver's input. This in turn means that we cannot correctly set the rewind security of the sub-protocols used in our final MPC protocol appropriately in advance.

We address this issue via the following strategy. We use *another* three round (delayed-input) extractable commitment scheme [34] (ecom) as well as another copy of promise ZK. This copy of promise ZK proves honest behavior in the first three rounds, and its third message is committed inside the extractable commitment. Further, the third round message of this extractable commitment is such that it allows for polynomial-time extraction (with the possibility of "over-extraction"[10]). This, however, comes at the cost that this extractable commitment does not achieve any rewind security. Interestingly, stand-alone security of this scheme suffices for our purposes since we only use it in the case where the adversary always aborts in the third round (and therefore, no rewinds are performed).

The main idea is that by using such a special-purpose extractable commitment scheme, we can ensure that an a priori fixed constant number of rewinds are sufficient for extracting the committed value, namely, the promise ZK third round prover message, with noticeable probability. This, in turn, allows us to set the rewind security of other sub-protocols used in our MPC protocol in advance to specific constants.

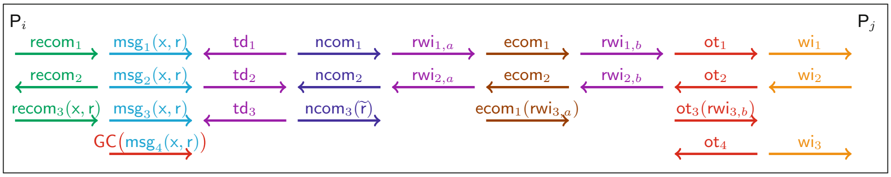
Of course, the adversary may always choose to commit to malformed promise ZK messages within the extractable commitment scheme. In this case, our simulator may always decide not to extract adversary's input, *even if the adversary was behaving honestly otherwise*. This obviously would lead to a view that is distinguishable from the real world. To address this issue, we use a *proofs of proofs* strategy. Namely, we require the first copy of promise ZK, which is encrypted inside OT, to prove that the second copy of promise ZK is "accepting". In this case, if the adversary commits malformed promise ZK messages within the extractable commitment, the promise ZK message inside OT will not be accepting. This, in turn, means that due to the security of garbled circuits, the fourth round messages of the parties will become "opaque".

Finally, we remark that for technical reasons, we do extract the promise ZK encrypted inside the OT receiver message in our *final* hybrid. However, in this particular hybrid, the number of rewinds required for extraction do not matter since in this hybrid, we only make change inside a *non-interactive* primitive (specifically, garbled circuit) that is trivially secure against an *unbounded* polynomial number of rewinds.

2.3 Protocol Design Summary

Putting all the various pieces together, we describe the overall structure of the protocol at a high level to demonstrate the purpose of its various components in the context of the protocol.

[10] This means the extractor can output a non $\perp$ value if the commitment has no valid opening.



For simplicity we consider the messages sent from P_i to P_j. Note that even though P_j is the intended recipient for the messages in a two party sub-protocol, the messages are broadcast to all parties.

Delayed Semi-malicious MPC (blue). P_i uses input x and randomness r to compute the messages msg_k for the bounded rewind secure four-round delayed semi malicious protocol Π.

Multiparty Conditional Disclosure of Secrets (red). As discussed earlier, the last message of Π is not sent in the clear but instead sent inside a garbled circuit GC used to implement MCDS. We use a four-round oblivious transfer protocol ot_k to allow the parties to obtain garbled circuit wire labels corresponding to their witnesses. We defer the discussion on the witness for MCDS below.

Rewind Secure Extractable commitment (green). The same input and randomness used to compute messages for Π is committed via an extractable commitment recom_k. This is done to enable the simulator to extract the inputs and randomness of the adversary for simulation. As discussed earlier, we use a three round extractable commitment that achieves bounded rewind security.

Promise ZK (purple). We use promise ZK in a non-black box manner in our protocol. Specifically, it consists of a trapdoor generation phase td_k, and a bounded rewind secure witness indistinguishable proof rwi_k. As discussed in our *proofs of proofs* strategy, we actually use two copies of the promise ZK (indexed by subscripts a and b in the figure), but both of these copies will share a single instance of the trapdoor generation. At a high level, both rwis prove that either the claim is true or "I committed to the trapdoor in the non-malleable commitment" (see below). We also note that one of the rwi copies, specifically, the copy indexed with subscript b is used as a witness for the MCDS mechanism.

Witness Indistinguishable Proof (orange). We also use a regular witness indistinguishable proof wi (without any rewind security) to establish honest behavior of the parties in the last round of the protocol. This effectively involves proving that either the last round message was computed honestly or "I committed to the trapdoor in the non-malleable commitment" (see below).

Extractable Commitment (brown). As discussed earlier, we use an extractable commitment ecom (without rewind security) to implement our *proofs of proofs* strategy to enable simulation.

Non-malleability (dark blue). We bootstrap non-malleability in our protocol using non-malleable commitments ncom in a similar manner to prior works [2,3]. Specifically, in the honest execution of the protocol, the parties simply commit to a random value $\widetilde{\mathsf{r}}$. We rely on specific properties of the ncom, which we do not discuss here and refer the reader to the technical sections.

Finally, we note that our protocol design uses multiple sub-protocols with bounded rewind security. We do not discuss how the bounds for the sub-protocols are set here, and instead defer this discussion to Sect. 5.

Complexity of the Protocol Description. One might wonder why our construction is so involved and whether there is a simpler construction. This is an important question that needs to be addressed. Unfortunately, our current understanding of the problem does not allow for a protocol that is easier to describe, but we believe that our solution is less complex than the prior state-of-the-art solutions [3,25].

3 Preliminaries

We present the syntax and informal definitions of some preliminaries below. Additional preliminaries, and the full definitions can be found in the full version.

3.1 Extractable Commitments with Bounded Rewinding Security

In this section, we describe an extractable commitment protocol that is additionally secure against a bounded number of rewinds. Since we are interested in the three round protocol, we limit our discussion in this section to this setting. A simple extractable commitment is a commitment protocol between a sender (with input x) and a receiver which allows an extractor, with the ability to rewind the sender via the second and third round of the protocol, to extract the sender's committed value. Several constructions of three round extractable commitment schemes are known in the literature (see, e.g., [34,35]).

When we additionally require bounded rewind security, we shall parameterize this bound by B_{recom}. Roughly this means that the value committed by a sender in an execution of the commitment protocol remains hidden even if a malicious receiver can rewind the sender back to the start of the second round of the protocol an a priori bounded B_{recom} number of times. Extraction will then necessarily require strictly larger than B_{recom} rewinds.

In the remainder of the section, we describe a construction of a three round extractable commitment protocol with bounded rewind security $\mathsf{RECom} = (S, R)$. The construction is adapted from the construction presented in [3], and simplified for our setting since we do not require the stronger notion of "reusability", as defined in their work.

In our application, we set $B_{\mathsf{recom}} = 4$; however, our construction also supports larger values of B_{recom}. For technical reasons, we don't define or prove B_{recom}-rewinding security property and reusability property for our extractable commitment protocol. Instead, this is done inline in our four round MPC protocol.

Construction. Let Com denote a non-interactive perfectly binding commitment scheme based on injective one-way functions. Let N and B_{recom} be positive integers such that $N - B_{\mathsf{recom}} - 1 \geq \frac{N}{2} + 1$. For $B_{\mathsf{recom}} = 4$, it suffices to set $N = 12$. The three round extractable commitment protocol RECom is described in Fig. 1.

Sender S has input x.

Commitment Phase:

1. **Round 1:** S does the following:
   - Pick N random degree B_{recom} polynomials $\mathsf{p}_1, \ldots, \mathsf{p}_N$ over $\mathbb{Z}_q$, where q is a prime larger than 2^λ.
   - Compute $\mathsf{recom}_{1,\ell}^{S\to R} \leftarrow \mathsf{Com}(\mathsf{p}_\ell; r_\ell)$ using a random string r_ℓ, for every $\ell \in [N]$.
   - Send $\mathsf{recom}_1^{S\to R} = (\mathsf{recom}_{1,1}^{S\to R}, \ldots, \mathsf{recom}_{1,N}^{S\to R})$ to R.
2. **Round 2:** R does the following:
   - Pick random values $\mathsf{z}_\ell \leftarrow_{\$} \mathbb{Z}_q$ for every $\ell \in [N]$.
   - Send $\mathsf{recom}_2^{R\to S} = (\mathsf{z}_1, \ldots, \mathsf{z}_N)$ to S.
3. **Round 3:** S does the following:
   - Compute $\mathsf{recom}_{3,\ell}^{S\to R} \leftarrow (x \oplus \mathsf{p}_\ell(0), \mathsf{p}_\ell(\mathsf{z}_\ell))$ for all $\ell \in [N]$.
   - Send $\mathsf{recom}_3^{S\to R} = (\mathsf{recom}_{3,1}^{S\to R}, \ldots, \mathsf{recom}_{3,N}^{S\to R})$ to R.

Decommitment Phase:

1. S outputs $\mathsf{p}_1, \ldots, \mathsf{p}_N$ together with the randomness $r_1, \ldots, r_N$ used in the first round commitments.
2. R first verifies the following:
   - For each $\ell \in [N]$, $\mathsf{recom}_{1,\ell}^{S\to R} = \mathsf{Com}(\mathsf{p}_\ell; r_\ell)$.
   - Parse $\mathsf{recom}_{3,\ell}^{S\to R} = (\alpha_\ell, \beta_\ell)$. Verify that $\beta_\ell = \mathsf{p}_\ell(\mathsf{z}_\ell)$.
   - For each $\ell \in [N]$, compute $x_\ell = \mathsf{p}_\ell(0) \oplus \alpha_\ell$. Verify that all the x_ℓ values are equal.

   If any of the above verifications fail, R outputs $\perp$. Otherwise, R outputs x.

Fig. 1. Extractable commitment scheme recom.

The corresponding properties for the above construction is presented in the full version.

3.2 Trapdoor Generation Protocol with Bounded Rewind Security

This section, we discuss the syntax and provide an intuitive definition, along with a sketched construction, for a Trapdoor Generation Protocol with Bounded Rewind Security [3]. The complete definition along with the construction is provided in the full version.

In a Trapdoor Generation Protocol, without bounded rewind security, a sender S (a.k.a. trapdoor generator) communicates with a receiver R. The protocol itself has no output, and the receiver has no input. The goal is for the sender to establish a trapdoor upon completion. On the one hand, the trapdoor can be extracted via a special extraction algorithm that has the ability to rewind the sender. On the other hand, no cheating receiver should be able to recover the trapdoor.

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 receiver R that proceeds as below.

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

In what follows, for brevity, we set td_1 to be $\mathsf{td}_1^{S \to R}$. Similarly we use td_2 and td_3 instead of $\mathsf{td}_2^{R \to S}$ and $\mathsf{td}_3^{S \to R}$, respectively. Note that the algorithm $\mathsf{TDValid}$ does not form a part of the interaction between the trapdoor generator and the receiver. It is, in fact, a public algorithm that enables public verification of whether a value t is a valid trapdoor for a first round message td_1.

The protocol satisfies two properties: (i) Sender security, i.e., no cheating PPT receiver can learn a valid trapdoor, and (ii) Extraction, i.e., there exists an expected PPT algorithm (a.k.a. extractor) that can extract a trapdoor from an adversarial sender via rewinding.

Extraction. There exists a PPT extractor algorithm TDExt that, given a set of values[11] $(\mathsf{td}_1, \{\mathsf{td}_2^i, \mathsf{td}_3^i\}_{i=1}^3)$ 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 t such that $\mathsf{TDValid}(\mathsf{t}, \mathsf{td}_1) = 1$.

1-Rewind Security. Intuitively, a Trapdoor Generation protocol is 1-rewind secure if it protects a sender against a (possibly cheating) receiver that has the ability to rewind it once. Specifically, the receiver is allowed to query the sender on two (possibly adaptive) different second round messages, thereby receiving two different third round responses from the sender. It should be the case that the trapdoor still remains hidden to the receiver.

Construction Based on One-way Functions. We sketch here the simple construction based on any signature scheme. In the first round, the sender samples a signing key pair and sends the verification key to the receiver. The receiver queries a random message in the second round, and the sender responds with the corresponding signature in the third. The trapdoor is defined to be 3 distinct (message, signature) pairs. It is easy to see that both extraction and 1-rewind security are satisfied for this construction.

[11] 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 feeds these values to the extractor TDExt.

3.3 Witness Indistinguishable Proofs with Bounded Rewinding Security

In this section we discuss the informal definition of a delayed input witness indistinguishable arguments (WI) to additionally satisfy B_{rwi}-bounded rewinding security, where the same statement is proven across all the rewinds. We refer to such primitives as B_{rwi}-bounded rewind secure WI.

B_{rwi}-Bounded Rewinding Security. The intuition for the definition is similar to that of the trapdoor generation protocol as described in the previous section. Here, for the three round delayed-input witness indistinguishable argument we want witness indistinguishability to be preserved as long as the verifier is restricted to rewinding the prover B_{rwi}-1 times. Specifically, the prover sends its first round message to the verifier, who then chooses (i) a triple consisting of a statement, and any two corresponding witnesses w_0 and w_1; (ii) B_{rwi}-1 s round verifier messages for the single first round prover message. The prover then completes the protocol, responding to each of the B_{rwi}-1 verifier messages, using either witness w_0 or w_1 for *every* response.

We refer the reader to the full version for the formal definitions of both, a delayed input WI, and the B_{rwi}-bounded rewind secure WI.

It was recently shown in [24] that there exists such WI arguments assuming non-interactive commitments. For further details, see the full version. We will use their scheme in our protocol.

3.4 Special Non-Malleable Commitments

In this work we make use of a commitment scheme that is non-malleable, non-malleable with respect to extraction and enjoys some additional properties. We refer to such a commitment scheme as a *special non-malleable commitment scheme*. We refer the reader to the full version for the basic definitions of non-malleable commitments. In the full version we also briefly detail the non-malleable commitment scheme of [22] and show that it is a special non-malleable commitment scheme.

4 Oblivious Transfer with Bounded Rewind Security

In this section we define, and then construct, a strengthening of regular oblivious transfer. We construct a rewinding secure Oblivious Transfer (OT) assuming the existence of four round OT protocol. For an OT protocol to be rewind secure, we require security against an adversary who is allowed to re-execute the second and third round of the protocol multiple times. But the first and fourth round are executed only once.

4.1 Definition

We start by formalizing the notion of a rewind secure oblivious transfer protocol. We shall denote by $\mathsf{out}_R\langle S(x), R(y)\rangle$ the output of the receiver R on execution of the protocol between R with input y, and sender S with input x. The four round oblivious transfer protocol is specified by four algorithms OT_j for $j \in [4]$; and the corresponding output protocol message is denoted by ot_j. We consider a delayed receiver input notion of the protocol where the receiver input is only required for the computation of ot_3.

Definition 1. *An interactive protocol (S, R) between a polynomial time sender S with inputs s_0, s_1 and polynomial time receiver R with input b, is a four round bounded rewind secure oblivious transfer (OT) if the following properties hold:*

- ***Correctness.*** *For any selection bit b, for any messages $s_0, s_1 \in \{0,1\}$, it holds that*

  $$\mathsf{Pr}\left[\mathsf{out}_R\langle S(s_0, s_1), R(b)\rangle = s_b\right] = 1$$

  where the probability is over the random coins of the sender S and receiver R.
- ***Security against Malicious Sender with B rewinds.*** *Here, we require indistinguishability security against a malicious sender where the receiver uses input $b[k]$ in the k-th rewound execution of the second and third round. Specifically, consider the experiment described below. $\forall \left\{b^0[k], b^1[k]\right\}_{k\in[B]} \in \{0,1\}$ where*

  Experiment E^{σ}**:**
  1. *Run OT_1 to obtain ot_1 which is independent of the receiver input. Send to $\mathcal{A}$.*
  2. *$\mathcal{A}$ then returns $\{\mathsf{ot}_2[j]\}_{j\in[B]}$ messages.*
  3. *For each $j \in [B]$, run OT_3 on $(\mathsf{ot}_1, \mathsf{ot}_2[j], b^{\sigma}[j])$ and send the response to $\mathcal{A}$.*
  4. *The output of the experiment is the entire transcript.*

  We say that the scheme is secure against malicious senders with B rewinds if the experiments E^0 and E^1 are indistinguishable.

4.2 Construction

We describe below the protocol Π^{R} which achieves rewind security against malicious senders. The Sender S's input is $s_0, s_1 \in \{0,1\}$ while the receiver R's input is $b \in \{0,1\}$.

Components. We require the following two components:

- $n \cdot \mathsf{B}_{\mathsf{OT}}$ instances of a 4 round OT protocol which achieves indistinguishability security against malicious senders.
- $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$ is a secure garbling scheme.

Protocol. The basic idea is to split the receiver input across multiple different OT executions such that during any rewind, a different set of OTs will be selected to proceed with the execution thereby preserving the security of the receiver's input. The sender constructs a garbled circuit which is used to internally recombine the various inputs shares and only return the appropriate output. The protocol is described below.

Round 1. (Π_1^{R}) : The receiver R computes the first round message of all the OTs. $\forall i \in [n], k \in [\mathsf{B}_{\mathsf{OT}}]$, $\mathsf{ot}_1^{i,k} := \mathsf{OT}_1\left(1^\lambda; \mathsf{r}_R\right)$ and send $\left\{\mathsf{ot}_1^{i,k}\right\}_{i \in [n], k \in [\mathsf{B}_{\mathsf{OT}}]}$ to S. We refer to index i as the outer index, and k as the inner index.

Round 2. (Π_2^{R}) : The sender S responds to all of the OT messages. $\forall i \in [n], k \in [\mathsf{B}_{\mathsf{OT}}]$, compute $\mathsf{ot}_2^{i,k} := \mathsf{OT}_2\left(\mathsf{ot}_1^{i,k}; \mathsf{r}_S\right)$ and sends $\left\{\mathsf{ot}_2^{i,k}\right\}_{i \in [n], k \in [\mathsf{B}_{\mathsf{OT}}]}$ to R.

Round 3. (Π_3^{R}) : The receiver now selects only a single OT to continue for i. It then encodes its input b by computing n additive shares and using each share as an input to a separate OT. Specifically, receiver R does the following:

- Compute n additive shares of b. Specifically, sample the first $n-1$ shares at random $\forall \ell \in [n-1]$ $b_\ell \leftarrow_{\$} \{0,1\}$ and set the last share $b_n := b \bigoplus_{\ell=1}^{n-1} b_j$.
- Sample within each tuple, the index for which to continue the OT. $\forall i \in [n]$, $\sigma_i \leftarrow_{\$} [\mathsf{B}_{\mathsf{OT}}]$.
- Use input b_i to compute the receiver message for $\mathsf{ot}_3^{i,\sigma_i}$. The other OTs are discontinued. Specifically, $\forall i \in [n]$, compute $\mathsf{ot}_3^{i,\sigma_i} \leftarrow \mathsf{OT}_3\left(b_i, \mathsf{ot}_1^{i,\sigma_i}, \mathsf{ot}_2^{i,\sigma_i}; \mathsf{r}_R\right)$ and send $\left\{\mathsf{ot}_3^{i,\sigma_i}, \sigma_i\right\}_{i \in [n]}$ to S.

Round 4. (Π_4^{R}) : The sender encodes its inputs (s_0, s_1) in a garbled circuit and uses the corresponding labels to complete the OT protocol.

- Compute garbled circuit: $\left(\overline{\mathsf{C}}_{\mathsf{ot}}, \overline{\mathsf{lab}}\right) := \mathsf{Garble}\left(\mathsf{C}_{\mathsf{ot}}\left[s_0, s_1\right]; \mathsf{r}_{\mathsf{gc},i}\right)$, where Circuit $\mathsf{C}_{\mathsf{ot}}[s_0, s_1]$ on input $b_1, \ldots, b_n$ outputs s_b where $b := \bigoplus_{i=1}^{n} b_i$.
- For $i \in [n]$, compute $\mathsf{ot}_4^{i,\sigma_i} := \mathsf{OT}_4\left(\mathsf{lab}_{i,0}, \mathsf{lab}_{i,1}, \mathsf{ot}_1^{i,\sigma_i}, \mathsf{ot}_2^{i,\sigma_i}, \mathsf{ot}_3^{i,\sigma_i}; \mathsf{r}_S\right)$ and send $\left\{\mathsf{ot}_4^{i,\sigma_i}\right\}_{i \in [n]}$ to R.

Evaluation. (OTEval') : The receiver R now evaluates the OT protocol to obtain labels needed to evaluate the output of the garbled circuit.

- For $i \in [n]$, compute $\widehat{\mathsf{lab}}_i := \mathsf{OTEval}\left(b_i, \mathsf{ot}_1^{i,\sigma_i}, \mathsf{ot}_2^{i,\sigma_i}, \mathsf{ot}_3^{i,\sigma_i}, \mathsf{ot}_4^{i,\sigma_i}; \mathsf{r}_R\right)$
- Output $s' := \mathsf{Eval}\left(\overline{\mathsf{C}}_{\mathsf{ot}}, \left\{\widehat{\mathsf{lab}}_i\right\}_{i \in [n]}\right)$

We now have the corresponding Lemma, which we prove in the full version.

Lemma 1. *Assuming receiver indistinguishability of* OT *against malicious senders, the receiver input in* Π^{R} *remains indistinguishable under* B_{OT}*-rewinds.*

Remark 1. We note that while our construction is proved against malicious senders, for our application it suffices to have the following two properties:

- bounded rewind security against semi malicious senders.
- standalone security against receivers.

Remark 2. While not relevant to the bounded rewind security of the scheme, we note that in our applications, a malicious sender might compute the garbled circuit incorrectly. This stems from the fact that there will be multiple participants evaluating the garbled circuit to compute the OT output. We will therefore have to prove that the messages of the protocol were in fact computed correctly.

4.3 Four Round Delayed Input Multiparty Computation with Bounded Rewind Security

Looking ahead, for our main result, we will compile an underlying semi-malicious protocol to achieve malicious security. In order to use the underlying semi-malicious protocol in a black-box manner, we will require the protocol to satisfy bounded rewind security. We start with an intuitive definition which we follow by formalizing the intuition.

To start with, we consider a four round delayed input semi-malicious protocols satisfying the following additional properties, where we denote by msg_k the messages of all parties output in the k-th round by Π.

1. **Property 1:** msg_1 and msg_2 of Π contain only messages of OT instances.
2. **Property 2:** msg_1 and msg_2 of Π do not depend on the input. The input is used only in the computation of msg_3 and msg_4.
3. **Property 3:** The simulator $\mathcal{S}$ simulates the honest parties' messages msg_1 and msg_2 via $\mathcal{S}_1$ and $\mathcal{S}_2$ by simply running the honest OT sender and receiver algorithms.
4. **Property 4:** msg_3 can be divided into two parts: (i) components independent of the OT messages; and (ii) OT messages.

Here we clarify what it means for a component of a message to be independent of OT messages. We say a component of msg_3 is independent of OT messages if its computation in the third round is independent of the both the private and public state of OT.

The recent works of [5,14] construct two round semi-malicious protocols. Both protocols when instantiated with a four round OT protocol, satisfy the above structure. This follows from the fact that when their protocols are instantiated with a four round OT protocol, the non-OT components of their protocol are executed only in round 3.

The bounded rewind security notion follows in similar vein to the bounded rewind secure primitives we have previously defined. Note that the primary difference here stems from the fact that the protocol we consider is in the simultaneous message model. We say that a protocol satisfying the above properties is bounded rewind secure if the protocol remains secure in the presence of an that adversary is able to rewind the honest parties in the second and third round of the execution. Specifically, an adversary is allowed to: (a) initially query $B - 1$ many distinct second round messages and receive third round messages in response; (b) in the last (B-th) query, the adversary also includes inputs for the honest parties. The adversary should then be unable to distinguish between the case that the protocol completes from the B-th query onward, where the last round was either completed with honest inputs provided by the adversary, or simulated.

Instantiation. On plugging in the bounded rewind-secure OT constructed in the previous section into the semi-malicious protocols of [5,14] gives us the required delayed input MPC protocol with bounded rewind security.

5 Four Round MPC

Building Blocks. We list below all the building blocks of our protocol.

- **Trapdoor Generation Protocol:** $\mathsf{TDGen} = (\mathsf{TDGen}_1, \mathsf{TDGen}_2, \mathsf{TDGen}_3, \mathsf{TDOut}, \mathsf{TDValid}, \mathsf{TDExt})$ is a three round B_{td}-rewind secure trapdoor generation protocol based on one-way functions (see Sect. 3.2). We set B_{td} to be 2.

  In our MPC construction, we use a "multi-receiver" version of TDGen that works as follows: whenever a sender party i sends its first round message td_1, *all* of the other $(n-1)$ parties send a second round receiver message $\mathsf{td}_{2,i}$. The sender now prepares $\mathsf{td}_2 = (\mathsf{td}_{2,1}||\ldots||\mathsf{td}_{2,n-1})$, and then uses it to compute td_3. All the $(n-1)$ receivers individually verify the validity of td_3.
- **Delayed-Input WI Argument:** $\mathsf{WI} = (\mathsf{WI}_1, \mathsf{WI}_2, \mathsf{WI}_3, \mathsf{WI}_4)$ is a three round delayed-input witness indistinguishable proof system (see Sect. 3.3), where WI_4 is used to compute the decision of the verifier.
- **Bounded-Rewind Secure WI Argument:** $\mathsf{RWI} = (\mathsf{RWI}_1, \mathsf{RWI}_2, \mathsf{RWI}_3, \mathsf{RWI}_4)$ is a three round delayed-input witness-indistinguishable proof with B_{rwi_a}-rewind security (see Sect. 3.3). RWI_4 is used to compute the decision of the verifier. We will use two different instances of RWI that we will refer to as RWI_a and RWI_b, where the subscripts a and b denote the different instances. We set their respective rewind security parameters B_{rwi_a} and B_{rwi_b} to be some fixed polynomial.
- **Special Non-malleable Commitment:** $\mathsf{NMCom} = \Big(\mathsf{NMCom}_1, \mathsf{NMCom}_2, \mathsf{NMCom}_3\Big)$ is a three round special non-malleable commitment scheme. Let $\mathsf{Ext}_{\mathsf{NMCom}}$ denote the extractor associated with NMCom.
- **Bounded-Rewind Secure Extractable Commitment:** $\mathsf{RECom} = (\mathsf{RECom}_1, \mathsf{RECom}_2, \mathsf{RECom}_3)$ is the three round B_{recom}-rewind secure delayed-input extractable commitment based on non-interactive commitments (see Section 3.1). We set rewinding security parameter B_{recom} to be 4. $\mathsf{Ext}_{\mathsf{RECom}}$ is the extractor associated with RECom.
- **Extractable Commitment:** $\mathsf{Ecom} = (\mathsf{Ecom}_1, \mathsf{Ecom}_2, \mathsf{Ecom}_3, \mathsf{Ext}_{\mathsf{Ecom}})$ is the three round delayed-input extractable commitment scheme based on statistically binding commitment schemes. They satisfy the 2-extraction property.
- **Delayed Semi-Malicious MPC:** Π is a four round B_{Π}-bounded rewind secure delayed input MPC protocol based on oblivious transfer (see Sect. 4.3). We set B_{Π} to be 9.
- **Garbled Circuits:** $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$ is a secure garbling scheme. We denote the labels $\{\mathsf{lab}_{i,0}, \mathsf{lab}_{i,1}\}_{i\in[L]}$ by $\overline{\mathsf{lab}}$. We will often partition the labels of the garbled circuit to indicate the party providing the input corresponding to the label indices, and denote this by $\overline{\mathsf{lab}}_{|_j}$ for party j.

- **Oblivious Transfer:** $\mathsf{OT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3, \mathsf{OT}_4)$ is a four round oblivious transfer protocol. We abuse notation slightly and use this as implementing parallel OT executions where the receiver's input is a string of length ℓ and the sender now has ℓ pairs of inputs. We require regular indistinguishability security against a malicious sender. In addition, we require extraction of the receiver's input bit.

Levels of Rewind Security. We recall the notion of bounded-rewind security and the need for levels of rewind security. Bounded-rewind security, as in [3], is used in the security proof to argue indistinguishability in intermediate hybrids. The main idea is that when arguing indistinguishability of two hybrids, to derive a contradiction it suffices to build an adversary with non-negligible success probability. As such, as long as the adversary does not abort with some non-negligible probability (which is indeed true), a small constant number of rewinds are sufficient for extracting with non-negligible probability. The exact bounded-rewind security constants for various primitives are carefully set to establish various "levels" of security.

For primitives with bounded rewind security, we require

$$B_{\mathsf{rwi}_a}, B_{\mathsf{rwi}_b}, B_{\Pi} > B_{\mathsf{recom}} > B_{\mathsf{td}}$$

where they denote the total number of rewinds (including the main thread) that they are secure against. In addition, we require all of them to be larger than the number of threads required to extract from NMCom and Ecom. For the above primitives, we have $B_{\mathsf{rwi}_a} = B_{\mathsf{rwi}_b} = \mathsf{poly}(\lambda)$ (for some fixed polynomial), $B_{\Pi} = 9$, $B_{\mathsf{recom}} = 4$ and $B_{\mathsf{td}} = 2$ thus satisfying our requirements.

Notation for Transcripts. We introduce a common notation that we shall use to denote *partial* transcripts of an execution of different protocols that we use in our MPC construction. For any execution of protocol X, we use $\mathbf{T}_X[\ell]$ to denote the transcript of the first ℓ rounds.

NP Languages. We define the NP languages used for the three different proof systems that we use in our protocol. We denote statements and witnesses as st and w, respectively.

1. RWI_a: We use RWI_a for language L_a, which is characterized by the following relation R_a:

$\mathsf{st} := \left(\mathbf{T}_{\Pi}[2], \left\{\mathbf{T}^{j}_{\mathsf{recom}}[3]\right\}_{j\in[n]}, \{\mathsf{msg}_\ell\}_{\ell\in[3]}, \mathbf{T}_{\mathsf{ncom}}[3], \mathsf{td}_1\right)$

$\mathsf{w} := \left(\mathsf{inp}, \mathsf{r}, \left\{\mathsf{r}^{j}_{\mathsf{recom}}\right\}_{j\in[n]}, \mathsf{t}, \mathsf{r}_{\mathsf{ncom}}\right)$

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

(a) **Honest:** *all* of the following conditions hold:
- $\forall j$, $\mathbf{T}^{j}_{\mathsf{recom}}[3]$ is a well-formed transcript of RECom w.r.t. input $(\mathsf{inp}, \mathsf{r})$ and randomness $\mathsf{r}^{j}_{\mathsf{recom}}$.
- for every $\ell \leq 3$, msg_ℓ is an honestly computed ℓ^{th} round message in the protocol Π w.r.t. input inp, randomness r and the first $(\ell - 1)$ round protocol transcript $\mathbf{T}_{\Pi}[\ell - 1]$.

(b) **Trapdoor:** $\mathbf{T}_{\mathsf{ncom}}[3]$ is an honest transcript of NMCom w.r.t. input t and randomness $\mathsf{r}_{\mathsf{ncom}}$ (AND) t is a valid trapdoor w.r.t. td_1

2. RWI_b: We use RWI_b for language L_b, which is characterized by the following relation R_b:

$\mathsf{st} := \left(\left\{\mathbf{T}^j_{\mathsf{rwi}_a}[2], \mathsf{st}^j_a, \mathbf{T}^j_{\mathsf{ecom}}[3]\right\}_{j\in[n]}, \mathbf{T}_{\mathsf{ncom}}[3], \mathsf{td}_1\right)$

$\mathsf{w} := \left(\left\{\mathsf{r}^j_{\mathsf{rwi}_a}, \mathsf{w}^j_a, \mathsf{rwi}^j_{3,a}, \mathsf{r}^j_{\mathsf{ecom}}\right\}_{j\in[n]}, \mathsf{t}, \mathsf{r}_{\mathsf{ncom}}\right)$.

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

(a) **Honest:** *all* of the following conditions hold:
   - $\forall j$, $\mathbf{T}^j_{\mathsf{ecom}}[3]$ is a well-formed transcript of Ecom w.r.t. input $\left\{\mathsf{rwi}^k_{3,a}\right\}_{k\in[n]}$ and randomness $\mathsf{r}^j_{\mathsf{ecom}}$.
   - $\forall j$, $\mathbf{T}^j_{\mathsf{rwi}_a}[2]||\mathsf{rwi}^j_{3,a}$ is an honestly computed transcript of RWI_a for L_a with statement st^j_a, witness w^j_a and randomness $\mathsf{r}^j_{\mathsf{rwi}_a}$. [a]

(b) **Trapdoor:** $\mathbf{T}_{\mathsf{ncom}}[3]$ is an honest transcript of NMCom w.r.t. input t and randomness $\mathsf{r}_{\mathsf{ncom}}$ (AND) t is a valid trapdoor w.r.t. td_1

[a] Since RWI is not publicly verifiable, the relation establishes that the RWI prover messages were computed honestly w.r.t. the witness and randomness for the statement.

3. WI: We use WI for language L_c, which is characterized by the following relation R_c:

$$\mathsf{st} := \Big(\{\mathsf{msg}_i, \mathsf{ncom}_i\}_{i\in[3]}, \left\{\mathsf{recom}^j_i\right\}_{i\in[3],j\in[n]}, \left\{\mathsf{rwi}^j_i\right\}_{i\in[2],j\in[n]}, \mathsf{Trans}_3, \left\{\mathsf{ot}^j_i\right\}_{i\in[4],j\in[n]}, \left\{\mathsf{st}^j\right\}_{j\in[n]}, \widetilde{\mathsf{C}}, \mathsf{td}_1, \mathbf{T}_{\mathsf{NMCom}}[3]\Big)$$

$\mathsf{w} := \left(\mathsf{inp}, \mathsf{r}, \left\{\mathsf{r}^j_{\mathsf{recom}}, \mathsf{r}^j_{\mathsf{ot}}, \mathsf{r}^j_{\mathsf{rwi}}\right\}_{j\in[n]}, \mathsf{msg}_4, \mathsf{r}_{\mathsf{gc}}, \mathsf{t}, \mathsf{r}_{\mathsf{ncom}}\right)$

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

(a) **Honest:** For every j, *all* of the following conditions hold:
   - $\mathbf{T}^j_{\mathsf{recom}}[3]$ is a well-formed transcript of RECom w.r.t. input $(\mathsf{inp}, \mathsf{r})$ and randomness $\mathsf{r}^j_{\mathsf{recom}}$.
   - msg_4 is honestly computed round 4 message of Π w.r.t. inp, randomness r and transcript $\mathbf{T}_{\Pi}[3]$.
   - $(\overline{\mathsf{C}}, \overline{\mathsf{lab}})$ is honest garbling of C that contains hardwired values msg_4, $\left\{\mathbf{T}^j_{\mathsf{rwi}}[2], \mathsf{st}^j_b, \mathsf{r}^j_{\mathsf{rwi}}\right\}_{j\in[n]}$, using randomness r_{gc}. (See Figure 2.)
   - ot^j_4 is honestly computed using $\overline{\mathsf{lab}}_{|_j}$, randomness $\mathsf{r}^j_{\mathsf{ot}}$ and transcript $\mathbf{T}^j_{\mathsf{ot}}[3]$. ($\mathbf{T}^j_{\mathsf{ot}}[4] = \mathbf{T}^j_{\mathsf{ot}}[3]||\mathsf{ot}^j_4$).

(b) **Trapdoor:** $\mathbf{T}_{\mathsf{ncom}}[3]$ is an honest transcript of NMCom w.r.t. input t and randomness $\mathsf{r}_{\mathsf{ncom}}$ (AND) t is a valid trapdoor w.r.t. td_1

$$\mathsf{C}\left[\mathsf{msg}_4, \left\{\mathbf{T}^j_{\mathsf{rwi}_b}[2], \mathsf{st}^j_b, \mathsf{r}^j_{\mathsf{rwi}_b}\right\}_{j\in[n]}\right]$$

Input: $\{\mathsf{rwi}^j_{3,b}\}_{j\in[n]}$

- If for every $j \neq i$, $\mathsf{RWI}_4\left(\mathsf{st}^j_b, \mathbf{T}^j_{\mathsf{rwi}_b}[2]\|\mathsf{rwi}^j_{3,b}; \mathsf{r}^j_{\mathsf{rwi}_b}\right) = 1$, **output** msg_4;
- Else, **output** $\perp$.

Fig. 2. Circuit C

5.1 The Protocol

In this section, we describe our four round MPC protocol between n players $\mathsf{P}_1, \cdots, \mathsf{P}_n$. Let x_i denote the input of party P_i. At the start of the protocol, each party samples a sufficiently long random tape to use in the various sub-protocols; let r_X denote the randomness used in sub-protocol X.

Notational Conventions. We establish some conventions for simplifying notation in the protocol description. We only indicate randomness as an explicit input for computing the first round message of a sub-protocol; for subsequent computations, we assume it to be an implicit input. Similarly, we assume that any next-message of a sub-protocol takes as input a partial transcript of the "previous" rounds, and do not write it explicitly. Whenever necessary, we augment our notation with superscript $i \rightarrow j$ to indicate the a instance of an execution of a sub-protocol between a "sender" i and "receiver" j (where sometimes, the sender is a prover and receiver is a verifier). When the specific instance is clear from context, we shall drop the superscript. When we wish to refer to multiple instances involving a party i, we will use the shorthand superscript $i \rightarrow \bullet$ or $\bullet \rightarrow i$, depending upon whether i is the sender or the receiver. For example, $\mathbf{T}^{i\rightarrow\bullet}_X[\ell]$ will be a shorthand to indicate $\left\{\mathbf{T}^{i\rightarrow j}_X[\ell]\right\}_{j\in[n]}$.

We will sometimes use explanatory comments within the protocol description, denoted as //comment. Finally, we note that all messages in the protocol are broadcast; if any party aborts during the first three rounds of the protocol, it broadcasts an abort in the subsequent round. We do not write this explicitly in the protocol, and assume it to be implicit. We now proceed to describe the protocol.

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

1. Delayed semi-malicious MPC Π: $\mathsf{msg}_{1,i} \leftarrow \Pi_1\left(\mathsf{r}_i\right)$.
2. Sender message of TDGen: $\mathsf{td}_{1,i} \leftarrow \mathsf{TDGen}_1\left(\mathsf{r}_{\mathsf{td},i}\right)$.

For every $j \neq i$:

3. Prover message of the three delayed-input WI argument systems
   - WI: $\mathsf{wi}_1^{i\to j} \leftarrow \mathsf{WI}_1(\mathsf{r}_{\mathsf{wi}}^{i\to j})$.
   - RWI_a: $\mathsf{rwi}_{a,1}^{i\to j} \leftarrow \mathsf{RWI}_1(\mathsf{r}_{\mathsf{rwi}_a}^{i\to j})$.
   - RWI_b: $\mathsf{rwi}_{b,1}^{i\to j} \leftarrow \mathsf{RWI}_1(\mathsf{r}_{\mathsf{rwi}_b}^{i\to j})$.

4. Sender message of the three delayed-input commitment schemes
   - Ecom: $\mathsf{ecom}_1^{i\to j} \leftarrow \mathsf{Ecom}_1(\mathsf{r}_{\mathsf{ecom}}^{i\to j})$.
   - RECom: $\mathsf{recom}_1^{i\to j} \leftarrow \mathsf{RECom}_1(\mathsf{r}_{\mathsf{recom}}^{i\to j})$.
   - NMCom: $\mathsf{ncom}_1^{i\to j} \leftarrow \mathsf{NMCom}_1(\mathsf{r}_{\mathsf{ncom}}^{i\to j})$.
5. Receiver message of OT: $\mathsf{ot}_1^{j\to i} \leftarrow \mathsf{OT}_1\left(\mathsf{r}_{\mathsf{ot}}^{j\to i}\right)$.

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

1. Delayed semi-malicious MPC Π: $\mathsf{msg}_{2,i} \leftarrow \Pi_2$.

For every $j \neq i$:

2. Receiver message of TDGen: $\mathsf{td}_2^{i\to j} \leftarrow \mathsf{TDGen}_2$.
3. Verifier message of the three delayed-input WI argument systems
   - WI: $\mathsf{wi}_2^{j\to i} \leftarrow \mathsf{WI}_2$
   - RWI_a: $\mathsf{rwi}_{a,2}^{j\to i} \leftarrow \mathsf{RWI}_2$
   - RWI_b: $\mathsf{rwi}_{b,2}^{j\to i} \leftarrow \mathsf{RWI}_2$
4. Receiver message of the three delayed-input commitment schemes
   - Ecom: $\mathsf{ecom}_2^{j\to i} \leftarrow \mathsf{Ecom}_2$.
   - RECom: $\mathsf{recom}_2^{j\to i} \leftarrow \mathsf{RECom}_2$.
   - NMCom: $\mathsf{ncom}_2^{j\to i} \leftarrow \mathsf{NMCom}_2$.
5. Sender message of OT: $\mathsf{ot}_2^{i\to j} \leftarrow \mathsf{OT}_2$.

Round 3: P_i computes and broadcasts the *third* round messages of the following protocols:

1. Delayed semi-malicious Π: $\mathsf{msg}_{3,i} \leftarrow \Pi_3\left(\mathsf{x}_i\right)$ using input x_i. //First step where P_i is using its input.
2. TDGen: $\mathsf{td}_{3,i} \leftarrow \mathsf{TDGen}_3$.

For every $j \neq i$:

3. NMCom: $\mathsf{ncom}_3^{i\to j} \leftarrow \mathsf{NMCom}_3(\widetilde{\mathsf{r}}_j)$ to commit to a random $\widetilde{\mathsf{r}}_j$.
4. RECom: $\mathsf{recom}_3^{i\to j} \leftarrow \mathsf{RECom}_3(\mathsf{x}_i, \mathsf{r}_i)$ to commit to $(\mathsf{x}_i, \mathsf{r}_i)$.
5. RWI: $\mathsf{rwi}_{a,3}^{i\to j} \leftarrow \mathsf{RWI}_3\left(\mathsf{st}_a^{i\to j}, \mathsf{w}_a^{i\to j}\right)$ to prove that $R_a(\mathsf{st}_a^{i\to j}, \mathsf{w}_a^{i\to j}) = 1$, where

$$\text{Statement } \mathsf{st}_a^{i\to j} := (\mathbf{T}_\Pi[2], \mathbf{T}_{\mathsf{recom}}^{i\to\bullet}[3], \{\mathsf{msg}_{\ell,i}\}_{\ell\in[3]}, \mathbf{T}_{\mathsf{ncom}}^{i\to j}[3], \mathsf{td}_{1,j})$$

$$\text{“Honest” witness } \mathsf{w}_a^{i\to j} := (\mathsf{x}_i, \mathsf{r}_i, \mathsf{r}_{\mathsf{recom}}^{i\to\bullet})$$

6. Ecom: $\mathsf{ecom}_3^{i\to j} \leftarrow \mathsf{Ecom}_3\left(\mathsf{rwi}_{a,3}^{i\to\bullet}\right)$ to commit to $\mathsf{rwi}_{a,3}^{i\to\bullet}$.
7. RWI_b: $\mathsf{rwi}_{b,3}^{i\to j} \leftarrow \mathsf{RWI}_3(\mathsf{st}_b^{i\to j}, \mathsf{w}_b^{i\to j})$ to prove that $R_b(\mathsf{st}_b^{i\to j}, \mathsf{w}_b^{i\to j}) = 1$, where

$$\text{Statement } \mathsf{st}_b^{i\to j} := (\mathbf{T}_{\mathsf{rwi}_a}^{i\to\bullet}[2], \mathsf{st}_a^{i\to\bullet}, \mathbf{T}_{\mathsf{ecom}}^{i\to\bullet}[3], \mathbf{T}_{\mathsf{ncom}}^{i\to j}[3], \mathsf{td}_{1,j})$$

$$\text{“Honest” witness } \mathsf{w}_b^{i\to j} := (\mathsf{r}_{\mathsf{rwi}_a}^{i\to\bullet}, \mathsf{w}_a^{i\to\bullet}, \mathsf{rwi}_{a,3}^{i\to\bullet}, \mathsf{r}_{\mathsf{ecom}}^{i\to\bullet})$$

8. OT: Receiver message $\mathsf{ot}_3^{j\to i} \leftarrow \mathsf{OT}_3(\mathsf{rwi}_{b,3}^{i\to j})$ using input $\mathsf{rwi}_{b,3}^{i\to j}$.

Round 4: P_i computes and broadcasts the following messages:

1. If $\exists j \neq i$ such that $\mathsf{TDValid}(\mathsf{td}_{1,j}, \mathsf{td}_{2,j}, \mathsf{td}_{3,j}) \neq 1$, **abort**.
   //where $\mathsf{td}_{2,j} := (\mathsf{td}_2^{1\to j}||\cdots||\mathsf{td}_2^{n\to j})$.
2. Delayed semi-malicious MPC Π: Fourth round message $\mathsf{msg}_{4,i} \leftarrow \Pi_4$.
3. Garbled Circuit: $\overline{\mathsf{C}}_i$, where $(\overline{\mathsf{C}}_i, \overline{\mathsf{lab}}_i) \leftarrow \mathsf{Garble}(\mathsf{C}[\mathsf{msg}_{4,i}, \mathbf{T}_{\mathsf{rwi}_b}^{\bullet\to i}[2], \mathsf{st}_b^{\bullet\to i}, \mathsf{r}_{\mathsf{rwi}_b}^{\bullet\to i}]; \mathsf{r}_{\mathsf{gc},i})$.
   Circuit C is defined in Figure 2.

For every $j \neq i$:

4. OT: Fourth round sender message $\mathsf{ot}_4^{i\to j} \leftarrow \mathsf{OT}_4\left(\overline{\mathsf{lab}}_{i|_j}\right)$ using input $\overline{\mathsf{lab}}_{i|_j}$
   //$\overline{\mathsf{lab}}_{i|_j}$ denotes labels corresponding to the input wires for P_j's input.
5. OT: Receiver randomness $\mathsf{r}_{\mathsf{ot}}^{j\to i}$. //This is used by other parties to compute OT output.
6. WI: $\mathsf{wi}_3^{i\to j} \leftarrow \mathsf{WI}_3\left(\mathsf{st}_c^{i\to j}, \mathsf{w}_c^{i\to j}\right)$, to prove that $R_c(\mathsf{st}_c^{i\to j}, \mathsf{w}_c^{i\to j}) = 1$, where
   Statement $\mathsf{st}_c^{i\to j} := (\mathbf{T}_\Pi[3], \mathbf{T}_{\mathsf{recom}}^{i\to\bullet}[3], \mathbf{T}_{\mathsf{rwi}_b}^{\bullet\to i}[2], \mathsf{st}_b^{\bullet\to i}, \mathbf{T}_{\mathsf{ot}}^{i\to\bullet}[4], \overline{\mathsf{C}}_i, \mathbf{T}_{\mathsf{ncom}}^{i\to j}[3], \mathsf{td}_{1,j})$
   "Honest" witness $\mathsf{w}_c^{i\to j} := (\mathsf{x}_i, \mathsf{r}_i, \mathsf{r}_{\mathsf{recom}}^{i\to\bullet}, \mathsf{r}_{\mathsf{ot}}^{i\to\bullet}, \mathsf{r}_{\mathsf{rwi}_b}^{\bullet\to i}, \mathsf{msg}_{4,i}, \mathsf{r}_{\mathsf{gc},i})$

Output Computation: P_i computes the following:

1. If $\exists j \neq i$, s.t. $\mathsf{WI}_4(\mathsf{st}_c^{j\to i}, \mathbf{T}_{\mathsf{wi}}^{j\to i}[3]) \neq 1$, output $\perp$ and **abort**.
2. Compute OT outputs: $\forall j \neq i, \forall k \neq \{i, j\}$,
   $\widehat{\mathsf{lab}}_{j|_k} \leftarrow \mathsf{OTEval}(\mathbf{T}_{\mathsf{ot}}^{j\to k}[4]; \mathsf{r}_{\mathsf{ot}}^{j\to k})$
3. Evaluate garbled circuits: $\forall j \neq i$, $\widehat{\mathsf{msg}}_{4,j} \leftarrow \mathsf{Eval}(\overline{\mathsf{C}}_j, \widehat{\mathsf{lab}}_j)$, where $\widehat{\mathsf{lab}}_j := (\widehat{\mathsf{lab}}_{j|_1}||\cdots||\widehat{\mathsf{lab}}_{j|_n})$.
   If any evaluation returns $\perp$, then output $\perp$ and **abort**.
4. Output $y_i \leftarrow \mathsf{OUT}(\mathsf{x}_i, \mathbf{T}_\Pi[4]; \mathsf{r}_i)$, where $\mathbf{T}_\Pi[4]$ includes $\mathbf{T}_\Pi[3]$ and $\widehat{\mathsf{msg}}_{4,j}$ for every j.

Our main result is stated in the following theorem.

Theorem 3. *Assuming the hiding property of oblivious transfer, the hiding property of extractable commitment, the hiding property of extractable commitment with bounded rewind security, delayed semi malicious protocol with bounded rewind security computing any function $\mathcal{F}$, special non-malleable commitments, witness indistinguishable proofs with bounded rewind security, security of garbled circuits, trapdoor generation protocol with bounded rewind security, in addition to the correctness of these primitives, then the presented protocol is a four round protocol for $\mathcal{F}$ secure against a malicious dishonest majority.*

Remark 3. All the above primitives can be based on one-way functions, non-interactive commitments and oblivious transfer (OT). In a recent note by Lombardi and Schaeffer [31], they give a construction of a perfectly binding non-interactive commitment based on perfectly correct key agreement. As they point out, such key agreement schemes can be based on perfectly correct oblivious transfer [15]. This gives us both a non-interactive commitment schemes, and one-way functions, based on perfectly correct oblivious transfer. Thus it suffices to instantiate all our primitives using just oblivious transfer.

We thus have the following corollary.

Corollary 1. *Assuming polynomially secure oblivious transfer with perfect correctness, our constructed protocol is a four round multiparty computation protocol for any function $\mathcal{F}$.*

The complete security analysis of the above protocol is presented in the full version. Below we present a high level description of the main ideas of the proof and how the bounded rewind-security parameters are set.

5.2 Security

We emphasize that our discussion below is informal, and not a complete picture of the simulator and hybrids. Our intent is to give an outline of the key hybrids and simulation steps to convey the main ideas. This will already highlight the need for various levels of rewind security, one of the main challenges in proving security. There are lots of other challenges that we do not discuss here, and similar to prior works, the full security analysis is much more complex and we refer the reader to the full version for the analysis.

One particular challenge that we ignore is that of an aborting adversary, either implicitly or explicitly, in the first three rounds of the protocol. The case of an explicitly aborting adversary is dealt with in a similar manner to [3,16] by initially sampling a partial transcript, using dummy inputs, to determine if the adversary aborts, and then re-sampling the transcript in case the adversary does not abort. For an implicitly aborting adversary, the simulator (via extraction) can determine if the adversary aborted, but honest parties are not aware of this in the first three rounds of the protocol. This case relies on the security of the multi-party CDS (via OT and garbled circuits) to deal with the implicit aborts. Stepping around these challenges, the main steps in the simulation involve (a) rewinding the adversary to extract the trapdoor and inputs; (b) completing the witness indistinguishable arguments using the extracted trapdoor; (c) simulating the underlying protocol using the output obtained from the ideal functionality.

Key Hybrid Components. We give below a high level overview of some key hybrids in keeping with our simplified description of the simulator above. This will allow us to discuss our specific choices for the level of rewinds.

- The first hybrid is identical to the real protocol execution. Each witness indistinguishable (WI) argument in our protocol allows for a *trapdoor witness*, arising from the trapdoor generation protocol and the non-malleable commitment (NMCom). We would like it to be the case that a simulator is able to derive the trapdoor and produce a simulated transcript via the *trapdoor witness*, an adversary should not be in possession of a *trapdoor witness* thereby forcing honest behavior if the witness indistinguishable argument is accepting.

  In order to argue that the adversary is not in possession of the *trapdoor witness*, we need to ensure the following *invariant*: the adversary does not commit to the trapdoor inside of the NMCom.

  In order to do so in this hybrid, we rely on the rewind security of the trapdoor generation protocol. Specifically, we extract from the NMCom by rewinding the adversary once in the second and third round (two total executions of the second and third round). If indeed the adversary was committing to the trapdoor, the extraction is successful with some noticeable probability and thereby breaking the rewind security of the trapdoor generation protocol. Note, as observed in [3], to arrive at a contradiction via reduction it is sufficient to extract with noticeable (as opposed to overwhelming) probability. This explains why we require $B_{\mathsf{td}} \geq 2$.

For each change that we subsequently make through the various primitives, we will bootstrap the above technique, and argue that this invariant continues to hold. Specifically, in order to arrive at a contradiction, we will extract from the NMCom to break the security property of the corresponding primitive if the *invariant* ceases to hold. This already gives us a flavor for primitives to be secure against (at least) two rewinds needed for the extraction from the NMCom.

- In this hybrid, the simulator creates sufficient rewind execution threads in order to extract the adversary's input and the trapdoors needed to prove the WI using the *trapdoor witness*. These rewind threads have the same first round messages as the "main" execution thread, but the second and third round messages are computed in each rewind thread with fresh randomness. The rewind threads terminate on completion of the third round of the protocol.
- In the previous hybrid, the simulator is still using the honest inputs in the rewind threads. In this hybrid the rewind threads are switched from using the honest party's inputs, to an honest execution with input 0. Note that these threads finish by the end of the third round.

  While the changes made in this hybrid are done in a sequence of steps, and needs to be argued carefully, the sequence closely resembles the changes that will be made in the main execution thread below. Therefore, we primarily focus on the hybrids pertaining to the main execution thread.
- In this hybrid, the simulator uses the trapdoors extracted from the rewind threads to commit to the trapdoor inside the NMCom on the main execution thread. In order to argue indistinguishability, we perform a reduction to an external NMCom challenger. In order to generate the transcript internally, and complete the reduction, we need to rewind the adversary to get the trapdoor and inputs. But this causes a problem since the rewind threads might require responses to challenges that are meant for the external challenger. Here, we rely on the fact that the third round of our instantiated NMCom has pseudorandom messages, allowing us to respond to adversarial queries in the third round, that cannot be forwarded to the external NMCom challenger. This prevents the need for bounded rewind security from the NMCom.
- In a sequence of sub-hybrids, the simulator uses the extracted trapdoor to complete both the bounded rewind secure witness indistinguishable arguments using the *trapdoor witness*. As seen above, for the reduction we will need to rewind the adversary to extract, thereby rewinding the external challenger. Since we require extraction of the adversary's inputs, the parameter for the bounded rewind secure witness indistinguishable argument needs to satisfy $B_{\mathsf{rwi}} > B_{\mathsf{recom}}$.
- In this hybrid, the simulator uses the extracted trapdoor to complete the witness indistinguishable argument. Since the third round of this protocol is completed in the fourth round of our compiled protocol, rewinding the adversary to extract the trapdoor and input in the second and third round circumvents issues discussed above. Therefore, we don't require this primitive to be rewind secure.
- In this hybrid, the simulator switches to committing to 0 inside the rewind secure extractable commitment (RECom). Unlike the previous cases, this is potentially circularity since the arguments above do not directly extend. This is because it cannot

be the case that the external challenger remains secure if we rewind the adversary B_{recom} times to extract its input.
Instead, this is argued carefully where initially we argue that switching to a commitment of a "junk" value in the third round of the RECom doesn't affect our ability to extract from the adversary. This "junk" commitment can be made without knowledge of any randomness of the specific RECom instance. To argue this, we rely on the bounded rewind security of the extractable commitment, while still extracting the trapdoor to complete the transcript. This gives us the requirement that $B_{\mathsf{recom}} > B_{\mathsf{td}}$. This then allows for extraction of input in the reduction without violating rewinding circularity since, on the look ahead threads to extract, we can commit to junk without affecting input extraction.

- In this hybrid, the simulator simulates the transcript of the underlying bounded rewind secure protocol Π. Here too, we require extracting the inputs in order to send it to the ideal functionality. Therefore, we require $B_{\Pi} > B_{\mathsf{recom}}$.

Acknowledgments. We would like to thank Alex Lombardi and Luke Schaeffer for pointing out to us that any OT with perfect completeness implies non-interactive commitment schemes, and, for suggesting to us to revise our theorem statements to reflect this observation. Vipul Goyal is supported in part by the NSF award 1916939, a gift from Ripple, a JP Morgan Faculty Fellowship, a PNC center for financial services innovation award, and a Cylab seed funding award. Arka Rai Choudhuri and Abhishek Jain are supported in part by DARPA/ARL Safeware Grant W911NF-15-C-0213, NSF CNS-1814919, NSF CAREER 1942789, Samsung Global Research Outreach award and Johns Hopkins University Catalyst award. Arka Rai Choudhuri is also supported by NSF Grants CNS-1908181, CNS-1414023, and the Office of Naval Research Grant N00014-19-1-2294. Rafail Ostrovsky is supported in part by NSF-BSF Grant 1619348, DARPA/SPAWAR N66001-15-C-4065, ODNI/IARPA 2019-1902070008 US-Israel BSF grant 2012366, JP Morgan Faculty Award, Google Faculty Research Award, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official views or policies, either expressed or implied, of the Department of Defense, DARPA, ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein. Michele Ciampi was partially supported by H2020 project PRIVILEDGE #780477.

References

1. Aiello, B., Ishai, Y., Reingold, O.: Priced oblivious transfer: how to sell digital goods. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 119–135. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_8
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. LNCS, vol. 10401, pp. 468–499. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_16
3. 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. LNCS, vol. 10992, pp. 459–487. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_16

4. 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. https://doi.org/10.1145/100216.100287
5. Benhamouda, F., Lin, H.: *k*-round multiparty computation from *k*-round oblivious transfer via garbled interactive circuits. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 500–532. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_17
6. Brakerski, Z., Halevi, S., Polychroniadou, A.: Four round secure computation without setup. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 645–677. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_22
7. Choudhuri, A.R., Ciampi, M., Goyal, V., Jain, A., Ostrovsky, R.: Round optimal secure multiparty computation from minimal assumptions. Cryptology ePrint Archive, Report 2019/216 (2019). https://eprint.iacr.org/2019/216
8. Ciampi, M., Ostrovsky, R., Siniscalchi, L., Visconti, I.: Delayed-input non-malleable zero knowledge and multi-party coin tossing in four rounds. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10677, pp. 711–742. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_24
9. 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. LNCS, vol. 10677, pp. 678–710. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70500-2_23
10. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). In: 23rd ACM STOC, pp. 542–552. ACM Press, May 1991. https://doi.org/10.1145/103418.103474
11. Dwork, C., Naor, M.: Zaps and their applications. In: 41st FOCS, pp. 283–293. IEEE Computer Society Press, November 2000. https://doi.org/10.1109/SFCS.2000.892117
12. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC, pp. 467–476. ACM Press, June 2013. https://doi.org/10.1145/2488608.2488667
13. Garg, S., Mukherjee, P., Pandey, O., Polychroniadou, A.: The exact round complexity of secure computation. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 448–476. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_16
14. Garg, S., Srinivasan, A.: Two-round multiparty secure computation from minimal assumptions. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 468–499. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78375-8_16
15. Gertner, Y., Kannan, S., Malkin, T., Reingold, O., Viswanathan, M.: The relationship between public key encryption and oblivious transfer. In: 41st FOCS, pp. 325–335. IEEE Computer Society Press, November 2000. https://doi.org/10.1109/SFCS.2000.892121
16. Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. J. Cryptol. **9**(3), 167–189 (1996). https://doi.org/10.1007/BF00208001
17. Goldreich, O., Krawczyk, H.: On the composition of zero-knowledge proof systems. SIAM J. Comput. **25**(1), 169–192 (1996). https://doi.org/10.1137/S0097539791220688
18. 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). https://doi.org/10.1145/28395.28420
19. Goldreich, O., Micali, S., Wigderson, A.: How to prove all NP statements in zero-knowledge and a methodology of cryptographic protocol design (extended abstract). In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 171–185. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_11
20. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989)

21. 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, June 2011. https://doi.org/10.1145/1993636.1993729
22. Goyal, V., Pandey, O., Richelson, S.: Textbook non-malleable commitments. In: Wichs, D., Mansour, Y. (eds.) 48th ACM STOC, pp. 1128–1141. ACM Press, June 2016. https://doi.org/10.1145/2897518.2897657
23. Goyal, V., Richelson, S.: Non-malleable commitments using Goldreich-Levin list decoding. In: Zuckerman, D. (ed.) 60th FOCS, pp. 686–699. IEEE Computer Society Press, November 2019. https://doi.org/10.1109/FOCS.2019.00047
24. Goyal, V., Richelson, S.: Non-malleable commitments using goldreich-levin list decoding. In: FOCS (2019)
25. Halevi, S., Hazay, C., Polychroniadou, A., Venkitasubramaniam, M.: Round-optimal secure multi-party computation. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10992, pp. 488–520. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_17
26. 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 (2008). https://doi.org/10.1007/978-3-540-85174-5_32
27. Jain, A., Kalai, Y.T., Khurana, D., Rothblum, R.: Distinguisher-dependent simulation in two rounds and its applications. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10402, pp. 158–189. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63715-0_6
28. Katz, J., Ostrovsky, R.: Round-optimal secure two-party computation. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 335–354. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_21
29. 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 (2003). https://doi.org/10.1007/3-540-39200-9_36
30. Kilian, J.: Founding cryptography on oblivious transfer. In: 20th ACM STOC, pp. 20–31. ACM Press, May 1988. https://doi.org/10.1145/62212.62215
31. Lombardi, A., Schaeffer, L.: A note on key agreement and non-interactive commitments. Cryptology ePrint Archive, Report 2019/279 (2019). https://eprint.iacr.org/2019/279
32. Pass, R.: Bounded-concurrent secure multi-party computation with a dishonest majority. In: Babai, L. (ed.) 36th ACM STOC, pp. 232–241. ACM Press, June 2004. https://doi.org/10.1145/1007352.1007393
33. 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 (2010). https://doi.org/10.1007/978-3-642-13190-5_32
34. Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: 43rd FOCS, pp. 366–375. IEEE Computer Society Press, November 2002. https://doi.org/10.1109/SFCS.2002.1181961
35. Rosen, A.: A note on constant-round zero-knowledge proofs for NP. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 191–202. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24638-1_11
36. Wee, H.: Black-box, round-efficient secure computation via non-malleability amplification. In: 51st FOCS, pp. 531–540. IEEE Computer Society Press, October 2010. https://doi.org/10.1109/FOCS.2010.87
37. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS, pp. 162–167. IEEE Computer Society Press, October 1986. https://doi.org/10.1109/SFCS.1986.25