Reusable Secure Computation in the Plain Model

Vipul Goyal^{*} Akshayaram Srinivasan[†] Mingyuan Wang[‡]

wingy dan wang

Consider the standard setting of two-party computation where the sender has a secret function f and the receiver has a secret input x and the output f(x) is delivered to the receiver at the end of the protocol. Let us consider the unidirectional message model where only one party speaks in each round. In this setting, Katz and Ostrovsky (Crypto 2004) showed that at least four rounds of interaction between the parties are needed in the plain model (i.e., no trusted setup) if the simulator uses the adversary in a black-box way (a.k.a. black-box simulation). Suppose the sender and the receiver would like to run multiple sequential iterations of the secure computation protocol on possibly different inputs. For each of these iterations, do the parties need to start the protocol from scratch and exchange four messages?

Abstract

In this work, we explore the possibility of *amortizing* the round complexity or in other words, *reusing* a certain number of rounds of the secure computation protocol in the plain model. We obtain the following results.

- Under standard cryptographic assumptions, we construct a four-round two-party computation protocol where (i) the first three rounds of the protocol could be reused an unbounded number of times if the receiver input remains the same and only the sender input changes, and (ii) the first two rounds of the protocol could be reused an unbounded number of times if the receiver input needs to change as well. In other words, the sender sends a single additional message if only its input changes, and in the other case, we need one message each from the receiver and the sender. The number of additional messages needed in each of the above two modes is optimal and, additionally, our protocol allows arbitrary interleaving of these two modes.
- We also extend these results to the multiparty setting (in the simultaneous message exchange model) and give round-optimal protocols such that (i) the first two rounds could be reused an unbounded number of times if the inputs of the parties need to change and (ii) the first three rounds could be reused an unbounded number of times if the inputs remain the same but the functionality to be computed changes. As in the two-party setting, we allow arbitrary interleaving of the above two modes of operation.

^{*}NTT Research and CMU vipul@cmu.edu

[†]Tata Institute of Fundamental Research akshayaram.srinivasan@tifr.res.in

[‡]UC Berkeley mingyuan@berkeley.edu

¹A. Srinivasan was supported in part by a SERB startup grant and Google India Research Award. M. Wang was supported in part by DARPA under Agreement No. HR00112020026, AFOSR Award FA9550-19-1-0200, NSF CNS Award 1936826, and research grants by the Sloan Foundation, and Visa Inc. Any opinions, findings and conclusions, or recommendations in this material are those of the authors and do not necessarily reflect the views of the United States Government or DARPA. This work was partly done when M. Wang was an intern at CMU.

Contents

1	Introduction 1.1 Our Results	3 3
2	Technical Overview 2.1 Reusable Two-Party Computation 2.1.1 Reusable Zero-Knowledge Argument of Knowledge 2.1.2 Reusable Oblivious Transfer Protocol 2.2 Reusable MPC	$5 \\ 6$
3	Preliminaries3.1Reusable Secure Two-Party Computation Protocol3.2Reusable Secure Multiparty Computation3.3Pseudorandom Generator, Pseudorandom Function, and Symmetric-key Encryption3.4Garbled Circuits3.5Extractable Commitment3.6ZAP3.7Trapdoor Generation Protocol	12 13 13 15
4	Reusable Zero-knowledge Argument of Knowledge 4.1 Construction 4.2 Proof of Zero-knowledge 4.3 Proof of Knowledge Extraction	18
5	Reusable Oblivious Transfer Protocol 5.1 A Building Block 5.2 Our Construction 5.3 Indistinguishability against a malicious sender. 5.4 Simulation security for a malicious receiver.	$\begin{array}{c} 23\\ 24 \end{array}$
6	Reusable 2PC Protocol6.1The Sender is Corrupt6.2The Receiver is Corrupt	
7	Reusable MPC Protocol 7.1 Additional Building Blocks 7.1.1 A Rewind-secure Extractable Commitment Scheme for Reusable MPC 7.1.2 (Reusable) Non-malleable Commitment. 7.1.3 Two-round (Reusable) Semi-Malicious MPC. 7.2 Our Protocol 7.3 The Simulator 7.4 The Hybrids 7.5 Indistinguishability of Hybrids.	32 33 34 34 38

1 Introduction

Secure computation [Yao86, GMW87] is a fundamental cryptographic primitive with numerous applications. A secure computation protocol allows a set of parties to compute a joint function of their private inputs while hiding everything about their inputs except the output of the functionality. This property is required to hold even against a centralized adversary that might corrupt any subset of the participating parties and instruct them to deviate arbitrarily from the protocol specification (known as malicious adversaries). In this work, we focus on constructing secure computation protocols that provide security against malicious adversaries in the *plain model* (i.e., without assuming any trusted setup).

Round Complexity of Secure Computation. Consider the standard setting of two-party computation where a sender and receiver are computing a certain function on their private inputs. Specifically, the receiver has a private input x and the sender has a secret function f. At the end of the protocol, the receiver learns f(x). We consider the unidirectional message model where, in each round of the protocol, a single party speaks. In this setting, Katz and Ostrovsky [KO04] showed that we need at least four rounds of communication in order to compute general functions if the simulator uses the adversary in a black-box way (i.e., black-box simulation). In the case of multiple parties, Garg et al. [GMPP16] showed a four-round lower bound in the simultaneous message exchange model where every party may speak in each round.

A long line of works, starting from the seminal works of Yao [Yao86] and Beaver, Micali, Rogaway [BMR90] have led to constructions of secure computation protocols that have minimal round complexity. Currently, we know of round-optimal malicious-secure protocols in the plain model in both the two-party and the multiparty setting under the minimal cryptographic hardness assumptions [KO04, IPS08, IKO⁺11, ORS15, GMPP16, BHP17, ACJ17, BL18, GS18, BGJ⁺18, HHPV18, FMV19, CCG⁺20, CCG⁺21].

Reusability of Rounds. Let us consider a setting where the sender and the receiver are not just interested in computing a single secure computation instance, but are interested in computing several instances sequentially possibly on different inputs. For example, say two banks collaborate on detecting if a large transaction is fraudulent. The banks might have an evolving database of past transactions. In this case, the parties participating in the 2PC remain the same while their input changes consistently. We ask the question of whether it is necessary for the sender and the receiver to start the protocol from scratch in each of these iterations and exchange four messages. Or, can they reuse some of the rounds of the protocol?

In this work, we begin the systematic study of reusable secure computation protocols in the plain model. Specifically, we study the possibility of constructing protocols where some of the rounds could be reused if the parties need to change either their private inputs or change the functionality to be securely computed. Prior to our work, reusable secure computation protocols were considered only in the common random string (CRS) model [BL20, BGMM20, AJJM20, AJJM21, BJKL21, BGSZ22], or in the plain model satisfying the weaker notion of super-polynomial simulation security [FJK21].

1.1 Our Results

In this work, we give round-optimal (i.e., four-round) constructions of two-party protocol (in the unidirectional message exchange model) and multiparty protocol (in the simultaneous message exchange model) for computing general functions where an *optimal* number of rounds could be reused an unbounded number of times.

Reusable 2PC. In the case of 2PC, we give a construction of a four-round protocol where (i) the first three rounds of the protocol can be reused an unbounded number of times if the receiver input remains the same and only the sender input changes and (ii) the first two rounds of the protocol can be reused an unbounded of times if the receiver input needs to change as well. We observe that the number of additional messages needed in each mode of reuse is optimal. First, any 2PC protocol in the unidirectional message model requires at least four rounds (w.r.t. black-box simulation) [KO04]. Furthermore, it is easy to see that, for every new sender input, at least one additional round is needed. Finally, if the receiver input needs to change, then we need at least two rounds of interaction. Otherwise, if there is only one more additional round, a semi-honest receiver could launch the so-called *residual attack* [HLP11] by evaluating it multiple

times with different choices of her input. Further, we allow the parties to interleave the above two modes of operation arbitrarily.² We prove the following theorem.

Theorem 1 (Reusable 2PC). Assume either DDH or QR assumption holds and the existence of a ZAP protocol. There exists a construction of a four-round reusable 2PC protocol with security against malicious adversaries.

We remark that in our model the sender maintains some state across each reuse session (i.e., in the sessions where the receiver's private input changes). This is used to ensure that the receiver's third-round messages across different reuse sessions are consistent. We argue that (refer to the technical overview) this is necessary for the unidirectional message model; without this, it is not possible to achieve even simple functionalities such as the zero-knowledge argument of knowledge.

Reusable MPC. In the case of multiple parties, we give a four-round protocol (in the simultaneous message exchange model) where (i) the first two rounds of the protocol can be reused an unbounded number of times if the parties need to change their private inputs and (ii) the first three rounds of the protocol can be reused an unbounded number of times if the private inputs remain the same but only the functionality to be computed changes. Again, as in the case of 2PC, we allow arbitrary interleaving of these two modes and observe that the number of additional messages needed in each reuse mode is optimal. We refer the reader to Section 3.2 for the formal definition of a reusable MPC. We prove the following theorem.

Theorem 2 (Reusable MPC). Assuming the existence of a four-round reusable 2PC protocol, a two-round semi-malicious reusable MPC protocol, and ZAPs. Then, there exists a construction of a four-round reusable MPC protocol with security against malicious adversaries.

We can instantiate the two-round semi-malicious reusable MPC protocol under the DDH assumption [BGMM20], SXDH assumption on asymmetric bilinear maps [BL20], LWE assumption [AJJM20, AJJM21, BJKL21], and the LPN assumption [BGSZ22].

Other Contributions. Along the way to obtaining our final results, we also construct a reusable zeroknowledge argument of knowledge (Section 4) and a reusable oblivious transfer (Section 5) in this paper. As far as we know, our work is the first to consider these reusable primitives in the plain model.

Alternate View of Our Results. Our protocols in the two-party and the multiparty setting have the following two-phase structure: in the first phase, we run a "mini" protocol to establish a reusable setup and in the second phase, we run a secure computation protocol that uses this setup. The number of rounds in the first and second phases depends on whether the private inputs of the parties change, or whether the functionality to be computed changes. In both settings, the number of rounds in the setup phase is at most 3. This must be contrasted with a naïve way of generating this setup which is to use a coin-tossing protocol to establish a CRS and then run a reusable 2PC/MPC in the CRS model. As shown in the works of Katz-Ostrovsky [KO04] and Garg et al. [GMPP16], such a coin-tossing protocol requires four rounds, and hence, this setup phase requires at least four rounds. In contrast, our approach leads to construction where at most 3 fixed and reusable rounds are required. In cases where only the function changes, we require 1 fresh round per execution. If the input of the parties also changes, we require 2 fresh rounds per execution. In addition, since our ultimate goal is to obtain a construction in the plain model, our constructions are arguably cleaner and simpler when compared to the coin-tossing-based approach.

2 Technical Overview

In this section, we give the key technical ideas behind our construction of a reusable two-party (in Section 2.1) and multiparty secure computation protocol (in Section 2.2).

 $^{^{2}}$ In particular, we refer to every new third-round message the receiver sends using a new input as a new *reuse session*. Within each reuse session, the sender could send multiple fourth-round messages using different inputs. By interleaving the two modes of reusability arbitrarily, we mean that the protocol execution could switch between reuse sessions (or create new reuse sessions) in an arbitrary manner. In fact, our protocol remains secure even if the adversary *adaptively* chooses which reuse session to execute next. We refer the reader to Section 3.1 for the formal definition of a reusable 2PC.

2.1 Reusable Two-Party Computation

As mentioned before, our goal is to construct a round-optimal (i.e., a four-round) two-party computation protocol where (i) the first three rounds of the protocol could be reused an unbounded number of times if the receiver input is fixed and only the sender input needs to change, and (ii) the first two rounds of the protocol could be reused an unbounded number of times if the receiver input needs to change as well.

The starting point of our construction is the (non-reusable) canonical two-party computation protocol in the plain model based on garbled circuits [Yao86]. Specifically, we consider Yao's 2PC protocol where the semi-honest oblivious transfer (OT) is replaced with a four-round malicious secure OT and the sender additionally uses a zero-knowledge argument of knowledge (ZK-AoK) to prove that the garbled circuit is generated correctly. The security of this protocol against malicious receivers follows directly from the zeroknowledge property of the underlying proof system, the security of the OT, and the security of garbled circuits. To give a bit more details, (i) we first use the simulator for the zero-knowledge proof to show that the proof gives no information about the witness (which is the secret randomness used to generate garbled circuits and the sender OT messages), (ii) we then use the simulator for the underlying OT protocol to extract the effective receiver input x, and (ii) finally, we use the security of garbled circuits to show that only the output f(x) is revealed. The security against malicious sender follows from the proof of knowledge property that allows the simulator to rewind and extract a witness proving that the garbled circuit was correctly generated. This witness is used to extract the effective sender input which is given to the ideal functionality.

We run into several roadblocks when we try to make this protocol reusable. The first issue is that the zero-knowledge property of the proof system might be completely compromised if the first three rounds of the protocol are reused for proving different statements. The second issue is that the receiver security of the OT protocol could be compromised if the first two rounds are reused (in the case when the receiver inputs need to change) and the sender security could be compromised if the first three rounds are reused (in the case when the sender inputs need to change). To get around these bottlenecks, we give constructions of round-optimal ZK-AoK and oblivious transfer protocol secure against malicious adversaries with the desired reusability properties. This forms the crux of our main technical contribution. Moreover, we show that the 2PC protocol constructed above inherits the reusability features of the underlying building blocks.

2.1.1 Reusable Zero-Knowledge Argument of Knowledge

Let us first try to construct a ZK-AoK protocol that runs in four rounds and the first three rounds of the protocol could be reused an unbounded number of times to prove different statements. This means that for each new statement, the prover needs to send a single message to the verifier and the verifier can check if the proof is accepting or rejecting. Zero-knowledge arguments for NP without this additional reusability feature are known from one-way functions [BJY97]. However, if we additionally require this reusability feature, then such a protocol can be shown to imply a pre-processing NIZK if the proof is not publicly verifiable and a designated prover NIZK when the proof is publicly verifiable. Indeed, the first three rounds of the protocol could be fixed as part of the CRS and the secret randomness used to generate the respective messages could be given to the prover and the verifier respectively as the proving and verifying keys. Now, the prover could use this key to generate proofs for an unbounded number of NP statements which could be verified by the verifier using the verifying key. All known constructions of pre-processing or designated prover NIZKs rely on assumptions stronger than one-way functions and, hence, it is likely that we need stronger assumptions to construct such a zero-knowledge protocol.

Our construction of reusable zero-knowledge builds on the FLS trapdoor paradigm [FLS90]. In this protocol, the first three rounds are used to generate a trapdoor between the verifier and the prover. In parallel, the prover and the verifier in rounds 2-4, run a delayed-input WI-PoK showing that either the statement is in the NP language or the prover knows the trapdoor. The trapdoor generation phase has the property that a cheating prover cannot extract a trapdoor whereas a rewinding simulator can extract it and use it to complete the WI argument. This allows us to argue PoK as well as zero-knowledge property. Unfortunately, this protocol as such is not reusable as the witness indistinguishability property of the WI protocol could completely break down if the first two rounds of the protocol are reused. This is indeed the case for the FLS delayed-input WI proof.

Our Solution. Our idea is to replace this delayed-input WI proof with a ZAP protocol [DN00]. ZAP is a two-round WI protocol between a verifier and a prover where the verifier's first-round message is a random string. Importantly, for our purposes, the same random string could be reused by the prover to prove multiple statements. Therefore, we can modify the above protocol so that the verifier in the third round also sends the first round message of a ZAP scheme. The prover in the final round proves via the ZAP that either the statement is true or it knows the trapdoor. This construction can be shown to be reusable zero-knowledge, meaning that the zero-knowledge property holds even if the first three rounds are reused. Unfortunately, it is not clear how to prove the soundness of this construction as ZAPs are not proofs of knowledge. To fix this, we additionally ask the prover to send an extractable commitment and show using ZAP that either statement is true or this extractable commitment is a valid commitment to the trapdoor. While this modification is sufficient to prove soundness, it is still not sufficient to show proof of knowledge property as ZAP does not allow extracting a valid witness from a malicious prover. To get around this problem, we ask the prover to send an additional extractable commitment to the witness and show via ZAP that either the first extractable commitment is a commitment to the trapdoor or the second extractable commitment is a commitment to a valid witness. An astute reader might have noticed that the extractable commitment to the witness must have the delayed-input property (meaning that the message is not determined until the end of the third round) and also have the first two rounds reusable. This means that using the same first two rounds of messages, a committer must be able to commit to a priori unbounded number of messages while maintaining the hiding property against a malicious receiver. We give a construction of such a commitment scheme in the main body. At a high level, the committer commits to a key of an SKE using a standard extractable commitment and sends an encryption of the message to be committed under the key in the final round. With this modification, we can show that the above protocol satisfies both reusable zero-knowledge (where the first three rounds are reused) as well as satisfies proof of knowledge property. We refer the reader to Section 4 for the details.

On the possibility of two-round reusable ZK-AoK. A natural question to ask is whether we can construct a round-optimal ZK-AoK protocol where the first two rounds are reused and the verifier can send arbitrary third-round messages in each reuse session. Indeed, if this ZK-AoK has to be used in our 2PC construction, then in each reuse session where the receiver's inputs change, a malicious receiver could choose an arbitrary third-round message and ask the prover to use this message to generate the final round proof. Unfortunately, we argue that such a ZK-AoK (with black-box simulators and extractors) cannot exist. This is because a malicious verifier could use the same strategy as that of the black-box extractor and generate multiple third-round messages and use the honest prover responses to extract a valid witness. The same argument also rules out the existence of a 2PC protocol for general functions where the receiver can choose arbitrary third-round messages. To get around this issue, we let the sender in the 2PC protocol maintain some state across the reuse sessions. Specifically, the sender could check if the third round message of the ZK-AoK protocol in each of the subsequent reuse sessions is the same as the ones used before. If this check does not pass, then the sender aborts. This check is necessary to get around the above-mentioned roadblock.

2.1.2 Reusable Oblivious Transfer Protocol

We now highlight the main technical challenges in constructing a reusable oblivious transfer protocol and explain how we overcome them. Recall that our goal is to construct a four-round (round-optimal) oblivious transfer protocol in the plain model that is secure against malicious adversaries and has first two round reusability (in case the receiver inputs need to change) and first three round reusability (in case only the sender inputs need to change).

We first observe that it is sufficient to construct an OT protocol with the above reusability features and satisfies standard simulation security against malicious receivers but only has indistinguishability-based security against malicious senders. Specifically, we require the view of a malicious sender to be computationally independent of the receiver's choice bits. Once we have such a protocol, we can upgrade its security by additionally asking the sender to give a ZK-AoK that it generated its OT messages correctly. If we rely on the ZK-AoK constructed in the previous subsection, then one could use the proof of knowledge extractor to extract the malicious sender inputs and thereby, show simulation security against corrupted senders. The zero-knowledge property of the proof system protects an honest sender against a malicious receiver. Additionally, this transformation preserves the desired reusability features of the underlying OT protocol. Hence, in the rest of this subsection, we focus on constructing a four-round OT protocol that has simulation security against malicious receivers and indistinguishability-based security against malicious senders.

Key Technical Challenge. The key technical challenge we face in constructing such an OT protocol is in designing an extractor that could extract the effective choice bit from a malicious receiver. This task is further complicated because (i) it must be accomplished within the first three rounds and hence, we cannot use any ZK-AoK to extract this information, and (ii) more importantly, we need the first two rounds of the protocol to be reusable an unbounded number of times. This means the honest receiver's input in each iteration must be hidden even if the first two round messages are fixed.

Starting Point. The starting point of our construction, as in the previous works [KO04, ORS15, FMV19], is a two-round semi-honest oblivious transfer from a special public-key encryption scheme. This special public-key encryption has the property that randomly sampled public keys are computationally indistinguishable from random strings (i.e., they are pseudorandom). Given such public-key encryption, the construction of a semi-honest oblivious transfer is as follows. The receiver chooses a valid public key pk_b (where b is its choice bit) and chooses a random string pk_{1-b} and sends (pk_0, pk_1) to the sender. The sender encrypts m_0 and m_1 under pk_0 and pk_1 respectively and sends it to the receiver. The receiver can now use the secret key sk_b corresponding to pk_b to decrypt and obtain m_b . In the semi-honest setting, we can set pk_{1-b} to be the same as the public key obtained from an external challenger and hence, one can use the semantic security of PKE to show that m_{1-b} is hidden. To make this construction secure against malicious receivers, the prior works [KO04, ORS15, FMV19] used a special kind of three-round commitment scheme called 1-outof-2 binding commitments. Specifically, in their construction, the receiver commits to two random strings via a standard extractable commitment. Let us call these two instances of the extractable commitment as $(Ecom_0, Ecom_1)$. In the second round, the sender sends a random string s. The receiver, in the third round, sends two strings (s_0, s_1) and proves using a WI proof that one of these strings is the same as the one that is committed in Ecom_0 or Ecom_1 . The honest receiver chooses s_{1-b} to be the same as the string committed in Ecom_0 and chooses $s_b = s \oplus pk$ where pk is a randomly sampled public key for which it knows the corresponding secret key. It completes the WI proof using s_{1-b} and the randomness used to generate Ecom_0 as the witness. The sender in the fourth round of the protocol sends encryptions of m_0 and m_1 under the public keys $s \oplus s_0$ and $s \oplus s_1$ respectively and the receiver can use the corresponding secret key to extract m_b .

To argue security against malicious receivers, we first observe that, from the soundness of the WI proof system, there exists some b such that s_{1-b} is the same as the value that was committed. The simulator first rewinds and extracts s_0, s_1 from the extractable commitment. It then sets s such that $s \oplus s_{1-b}$ is the public key obtained from the external challenger.³ This allows us to argue that m_{1-b} is hidden and formally proves security against malicious receivers. Indistinguishability against malicious senders is shown via a careful hybrid argument.

Challenges in the Reusable Setting. Unfortunately, the above construction is completely insecure in the reusable setting. Namely, even if the first two messages of the protocol are reused once, then a corrupt sender can look at the two strings sent by the honest receiver in the third round and figure out if the choice bits used in the two iterations are the same or different. To get around this issue, we could try to use a delayed-input extractable commitment where the first two messages are reusable and the message to be committed is known only in the third round (see the previous discussion for such a construction). In this case, the construction becomes trivially insecure against a malicious receiver. This is because the sender sends s in the second round and this is fixed in all the executions. Therefore, the receiver could choose (s_0, s_1) such that it knows the corresponding secret keys for both $s \oplus s_0$ and $s \oplus s_1$ and break the sender's privacy.

³Here, the simulator needs to first guess the value of the malicious receiver's choice bit b and set s accordingly. In the third round, it checks if the guessed value is correct and proceeds only in that case.

Our Solution. Observe that in the non-reusable version, security against corrupted receivers crucially relied on sampling the string s uniformly after the receiver sent the commitments. However, in the reusable setting, the string s is "chosen once for all" and fixed in all the subsequent executions. Thus, to make the above template reusable secure, we need to come up with a mechanism wherein we can "derandomize" the choice of s so that a single fixed s works for every execution. Towards this purpose, we do the following:

- 1. Let us start with the flawed approach discussed earlier where s_0, s_1 are committed using a reusable delayed-input extractable commitment. In this case, the receiver can trivially break the security by choosing s_0 and s_1 such that $s \oplus s_c$ for $c \in \{0, 1\}$ is a valid public key for which it knows the corresponding secret key. In hindsight, the insecurity of this construction stems from the fact that the receiver has complete control in choosing the public keys for both branches 0 and 1. As a first step, we modify the construction so that the receiver has full power to choose the public keys for one of the branches (namely, the branch corresponding to its choice bit b) but the possible choices of the public keys for the other branch are restricted, meaning that the number of possible choices of the string s_{1-b} that a corrupt receiver can choose is exponentially small.
- 2. Though the first step makes progress in restricting the power of a malicious receiver, we are still not done. This is because if the receiver is able to find a valid public key pk such that $pk \oplus s$ is in the restricted set, then it can use it to recover both the sender inputs. For instance, if the set of valid public keys is dense (as in the case of El-Gamal encryption), then the receiver can still break the security of the OT protocol. Hence, our next step is to use public-key encryption that has pseudorandom public keys, and additionally, the set of valid public keys is sparse. For security, we require that if a message is encrypted under an invalid public key, then it is statistically hidden. In the technical section (Section 5.1), we give constructions of such public-key encryption from standard assumptions such as DDH and QR. Given such public-key encryption, we can argue the security of our OT protocol against malicious receivers as follows. Since the number of possible choices of s_{1-b} is exponentially small and since the set of valid public keys is sparse, the set of possible values of s such that $s = s_{1-b} \oplus pk$ is exponentially small. Therefore, a randomly chosen s with overwhelming probability does not belong to this bad set and thus, can be fixed once and for all.

We now explain both the steps in a bit more detail. The approach we take in step-1 is inspired by Naor's commitments [Nao91]. Specifically, we use an extractable commitment to commit to two random strings as before, but instead of showing that one of (s_0, s_1) is the same as the value committed in $Ecom_0$ or $Ecom_1$, we show that one of (s_0, s_1) is equal to a PRG applied on the value that is committed in $Ecom_0$ or $Ecom_1$. Since the set of valid public keys is sparse, with overwhelming probability a randomly chosen s has the property that the set $\{s \oplus PRG(\cdot)\}$ has zero intersection with the set of valid public keys (if the PRG has a sufficiently large stretch). This allows us to argue that even if the receiver has the power to choose the value inside the extractable commitment after seeing s, it cannot break the sender's privacy. Furthermore, we observe that a similar extraction strategy as explained earlier allows us to extract the effective choice bit from the malicious receiver and prove simulation security.

To argue receiver privacy, we design a careful hybrid argument wherein we use the security of the underlying primitives. Specifically, we first switch s_b to also be the output of a PRG on a randomly chosen seed r_b by relying on the pseudorandomness of the public keys and that of the PRG. We commit to r_b in Ecom_1 and then switch the WI proof to using the randomness corresponding to Ecom_1 and r_b as the witness. We then switch Ecom_0 to be a commitment of r_b and switch $s_{1-b} = s \oplus pk$ for a randomly chosen pk. We now reverse the WI proof to use the randomness in Ecom_0 as the witness and switch Ecom_1 to be a commitment of a random seed. The final hybrid corresponds to an honest execution where the receiver's choice bit is 1 - b.

Getting 3-round Reusability. The above approach allows us to construct an oblivious transfer protocol where the first two rounds could be reused an unbounded number of times if the receiver inputs need to change. To make the first three rounds of the protocol to be reusable, we do the following. Instead of using (m_0, m_1) as the sender OT inputs, we sample two random keys (k_0, k_1) of an SKE scheme and use them as the sender OT inputs. In the final round, we send encryption of m_0 under k_0 and encryption of m_1 under k_1 . If the receiver input b remains the same, then in each iteration, the receiver only learns k_b and k_{1-b} is hidden. It now follows from the security of the SKE scheme that m_{1-b} in each iteration is hidden and hence, this modification can be shown to have the desired reusability features.

2.2 Reusable MPC

In a recent work, Choudhuri et. al. [CCG⁺20] constructed a four-round malicious-secure MPC protocol from the minimal assumption that a four-round malicious-secure OT protocol exists. In particular, their protocol makes use of several building blocks such as a four-round OT protocol, a rewind-secure WI proof, a rewindsecure semi-malicious four-round MPC protocol, etc. Via a careful parallelization of these building blocks, they show how to obtain a four-round malicious-secure MPC protocol.

Similar to the reusable 2PC case, we observe that if we instantiate every building block with a variant that supports reusability, then the protocol shall also be reusable. In particular, we shall instantiate the four-round OT protocol with the reusable OT protocol that we constructed earlier. The rewind-secure WI proof shall be replaced by the ZAP protocol (which naturally satisfies rewinding security). Moreover, we instantiate the four-round semi-malicious MPC protocol with the two-round reusable semi-malicious MPC protocol [BGMM20, BL20, AJJM20, AJJM21, BJKL21] where the first round message could be fixed and one can evaluate multiple functions by sending a single final round message. We prove that by using these reusable variants of the building blocks, the construction of Choudhuri et al. [CCG⁺20] is reusable. We refer the reader to Section 7 for the formal description.

3 Preliminaries

Let λ be the security parameter. We use $\operatorname{negl}(\lambda)$ to denote a negligible function. That is, for all polynomial $p(\lambda)$, it holds that $\operatorname{negl}(\lambda) < 1/p(\lambda)$ for large enough λ . For a randomized function f, we use f(x;r) to denote the evaluation of f with input x and randomness r. For any distribution A, we use $a \leftarrow A$ to denote that a is drawn according to distribution A. For any two distributions A and B, we use $A \stackrel{\sim}{\approx} B$ to denote that any PPT distinguisher \mathcal{D} cannot distinguish A and B with non-negligible advantage, i.e., $\left|\Pr[\mathcal{D}(1^{\lambda}, A) = 1] - \Pr[\mathcal{D}(1^{\lambda}, B) = 1]\right| = \operatorname{negl}(\lambda)$. For any positive integer n, we use [n] to denote the set $\{1, 2, \ldots, n\}$.

3.1 Reusable Secure Two-Party Computation Protocol

We consider the standard setting of two-party computation in the unidirectional message model. Specifically, the receiver has a private input a string x and the sender has a private input a function f. At the end of the protocol, the receiver learns f(x) and the sender gets no output. By the unidirectional message model, we refer to the setting where only one party speaks in each round of the protocol. We are interested in constructing round-optimal (i.e., four-round) protocols for this task.

Syntax. A four round reusable secure two-party computation protocol is given by a set of algorithms $\Pi = (\Pi_1, \Pi_2, \Pi_3, \Pi_4, \mathsf{out}_{\Pi})$ with the following syntax.

- $\Pi_1(1^{\lambda}, 1^{|x|})$: It is a PPT algorithm that is run by the receiver and it outputs the first round message π_1 and a secret receiver state st_R .
- $\Pi_2(1^{\lambda}, \pi_1, 1^{|f|})$: It is a PPT algorithm that is run by the sender and outputs the second round message π_2 and a secret sender state st_S .
- $\Pi_3(1^{\lambda}, \pi_2, x, \mathsf{st}_R)$: It is a PPT algorithm that is run by the receiver and outputs the third round message π_3 .
- $\Pi_4(1^{\lambda}, \pi_3, f, \mathsf{st}_S)$: It is a PPT algorithm that is run by the sender and outputs the final round message π_4 and outputs the updated sender state st_S .
- $\operatorname{out}_{\Pi}(\pi_3, \pi_4, x, \operatorname{st}_R)$: It is a deterministic algorithm that is run by the receiver and provides the output y.

Remark 1. As mentioned, for the unidirectional message model, it is necessary that Π_4 outputs an updated sender state to get around the impossibility of constructing reusable ZK-AoK w.r.t. black-box simulator and extractor.

Definition 1 (Reusable Security against Corrupted Receivers). We say that the protocol Π satisfies reusable security against corrupted receivers if for every non-uniform (stateful) PPT adversary \mathcal{A} that is corrupting the receiver, there exists an expected PPT (stateful) simulator Sim such that for all $\ell = \operatorname{poly}(\lambda)$ and for all non-uniform (stateful) PPT distinguishers D, the output of the real and ideal executions defined below are computationally indistinguishable.

- Real Execution:
 - 1. A on input $(1^{\lambda}, 1^{\ell})$ executes the first two rounds of the protocol with the honest sender.
 - 2. In the *i*-th reuse session:
 - (a) A either sends a fresh third round message which is forwarded to the sender or asks the sender to reuse a prior third round message.
 - (b) D on input the current view of A outputs the honest sender input $f^{(i)}$ for this session.
 - (c) The honest sender generates the final round message in the protocol using the input $f^{(i)}$ and the adversarially chosen third round message (which is either fresh or reused from a prior session). This final round message is forwarded to the adversary.
 - 3. The output of the real execution corresponds to the final output of D which is given the view of A at the end of the ℓ -th reuse session.
- Ideal Execution:
 - 1. Sim on input $(1^{\lambda}, 1^{\ell})$ and oracle access to the adversary \mathcal{A} generates the view of the adversary at the end of the first two rounds and outputs this view.
 - 2. The ideal execution starts by initializing the adversary \mathcal{A} with this view and continues with the rest of the execution as follows.
 - 3. In the *i*-th reuse session:
 - (a) \mathcal{A} either sends a fresh third round message or asks the sender to reuse a prior third round message.
 - (b) D on input the current view of A outputs the honest sender input $f^{(i)}$ for this session.
 - (c) The simulator on input the relevant third round message from the adversary outputs $x^{(i)}$.
 - (d) The simulator is provided with $f^{(i)}(x^{(i)})$ and uses it to generate the final round message in the protocol. This message is forwarded to the adversary.
 - 4. The output of the ideal execution corresponds to the final output of D which is given the view of A at the end of the ℓ -th reuse session.

Definition 2 (Reusable Security against Corrupted Senders). We say that the protocol Π satisfies reusable security against corrupted senders if for every non-uniform (stateful) PPT adversary \mathcal{A} that is corrupting the sender there exists an expected PPT (stateful) simulator Sim such that for any $\ell = \operatorname{poly}(\lambda)$ and for all non-uniform (stateful) PPT distinguishers D, the output of the real and ideal executions defined below are computationally indistinguishable.

• Real Execution:

- 1. A on input $(1^{\lambda}, 1^{\ell})$ executes the first two rounds of the protocol with the honest receiver.
- 2. In the *i*-th reuse session:
 - (a) \mathcal{A} either asks the honest receiver to reuse the third round message in the *j*-th reuse session (for some j < i) or asks it to generate a fresh third round message.
 - (b) In the case where a fresh message is requested, D on input the current view of A outputs $x^{(i)}$. This input is used by the honest receiver to sample a fresh third-round message.

- (c) The adversary generates a final round message in the protocol. The honest receiver computes the output of the protocol and this output is forwarded to the adversary.
- 3. The output of the real execution corresponds to the final output of D which is given the view of A at the end of the ℓ -th reuse session.
- Ideal Execution:
 - 1. Sim on input $(1^{\lambda}, 1^{\ell})$ and oracle access to the adversary \mathcal{A} generates the view of the adversary at the end of the first two rounds and outputs this view.
 - 2. The ideal execution starts by initializing the adversary \mathcal{A} with this view and continues with the rest of the execution as follows.
 - 3. In the *i*-th reuse session:
 - (a) \mathcal{A} either asks to reuse the third round message in *j*-th reuse session or asks to sample a fresh third round message. In the former case, we reset $x^{(i)} = x^{(j)}$. In the latter case, D on input the current view of \mathcal{A} outputs $x^{(i)}$.
 - (b) Depending on \mathcal{A} 's request, Sim either reuses the third round message in the *j*-th session or samples a fresh third round message (without the knowledge of $x^{(i)}$).
 - (c) The adversary generates a final round message in the protocol which is forwarded to the simulator. The simulator either provides $f^{(i)}$ which is forwarded to the trusted functionality or instructs the trusted functionality to output \perp . In the former case, the output of the honest receiver is set to $f^{(i)}(x^{(i)})$ and in the latter case, it is set to \perp . This output is forwarded to the adversary.
 - 4. The output of the ideal execution corresponds to the final output of D which is given the view of A at the end of the ℓ -th reuse session.

3.2 Reusable Secure Multiparty Computation

We consider the setting of multiparty computation in the bidirectional message model. Specifically, all parties hold an input x_i . Given a function f to compute, at the end of the protocol, all parties learn $f(x_1, \ldots, x_n)$. By bidirectional message model, we refer to the setting where all parties speak in each round of the protocol. We are interested in constructing round-optimal (i.e., four-round) protocols for this task.

Syntax. A four round reusable secure multiparty computation protocol is given by a set of algorithms $\Pi = (\Pi_1, \Pi_2, \Pi_3, \Pi_4, \mathsf{out}_{\Pi})$ with the following syntax.

- $\Pi_1(1^{\lambda}, 1^{|x|})$: It is a PPT algorithm that is run by the party to generate the first round message π_1 and a secret state st.
- $\Pi_2(1^{\lambda}, \pi_1, 1^{|f|}, st)$: It is a PPT algorithm that is run by the party to generate the second round message π_2 and an updated state st.
- $\Pi_3(1^{\lambda}, \pi_2, x, st)$: It is a PPT algorithm that is run by the party to generate the third round message π_3 .
- $\Pi_4(1^{\lambda}, \pi_3, f, st)$: It is a PPT algorithm that is run by the party to generate the final round message π_4 and an updated state st.
- $\operatorname{out}_{\Pi}(\pi_3, \pi_4, x, \operatorname{st})$: It is a deterministic algorithm that is run by the party and provides the output y.

Definition 3 (Reusable Security). We say that the protocol Π satisfies reusable security if for every nonuniform (stateful) PPT adversary \mathcal{A} that is corrupting a subset of parties $\mathcal{I} \subset [n]$ (let \mathcal{H} be the set of honest parties), there exists an expected PPT (stateful) simulator Sim such that for all $\ell = \operatorname{poly}(\lambda)$ and for all non-uniform (stateful) PPT distinguishers D, the output of the real and ideal executions defined below are computationally indistinguishable.

• Real Execution:

- 1. A on input $(1^{\lambda}, 1^{\ell})$ executes the first two rounds of the protocol with the honest parties.
- 2. In the *i*-th reuse session:
 - (a) \mathcal{A} either asks the honest parties to reuse the third round message in the *j*-th reuse session (for some j < i) or requests to compute a new reuse session.
 - (b) In the case where a fresh session is requested, D on input the current view of \mathcal{A} outputs the honest input $\{x_k^{(i)}\}_{k\in\mathcal{H}}$ for the honest parties. \mathcal{A} and the honest parties with this input execute the third round of the protocol.
 - (c) D, on input the current view of the adversary, output a function $f^{(i)}$ to be computed in the fourth round. A executes the final round message with the honest parties computing the function $f^{(i)}$. The output of the honest parties is forwarded to A.
- 3. The output of the real execution corresponds to the final output of D which is given the view of A at the end of the ℓ -th reuse session.

• Ideal Execution:

- 1. Sim on input $(1^{\lambda}, 1^{\ell})$ and oracle access to the adversary \mathcal{A} generates the view of the adversary at the end of the first two rounds and outputs this view.
- 2. The ideal execution starts by initializing the adversary \mathcal{A} with this view and continues with the rest of the execution as follows.
- 3. In the *i*-th reuse session:
 - (a) \mathcal{A} either asks to reuse the third round message in j-th reuse session or asks to sample a fresh third round message. In the former case, the party's input is reset as $x_k^{(i)} = x_k^{(j)}$ for each $k \in \mathcal{H}$. In the latter case, D on input the current view of \mathcal{A} outputs the party's input $\{x_k^{(i)}\}_{k\in\mathcal{H}}$. On receiving the third round message from \mathcal{A} , Sim outputs $\{x_k^{(i)}\}_{k\in\mathcal{I}}$.
 - (b) Depending on \mathcal{A} 's request, Sim either reuses the third round message in the *j*-th session or samples a fresh view for the third round (without the knowledge of $x^{(i)}$).
 - (c) The distinguisher D on the current view of the adversary outputs a function $f^{(i)}$ to compute. The simulator outputs $(f^{(i)}, \{x_k^{(i)}\}_{k\in\mathcal{I}})$ which is forwarded to the trusted functionality. The trusted functionality replies with $f^{(i)}(x_1^{(i)}, \ldots, x_n^{(i)})$. The simulator instructs the trusted functionality to deliver \perp or the output to the honest parties. In the latter case, the output of the honest party is set to $f^{(i)}(x_1^{(i)}, \ldots, x_n^{(i)})$ and in the former case, it is set to \perp . The output of the honest parties is forwarded to \mathcal{A} .
- 4. The output of the ideal execution corresponds to the final output of D which is given the view of A at the end of the ℓ -th reuse session.

3.3 Pseudorandom Generator, Pseudorandom Function, and Symmetric-key Encryption

Definition 4 (PRG). A function PRG: $\{0,1\}^{\lambda} \to \{0,1\}^{m}$, where $m > \lambda$, is called a pseudorandom generator if for all PPT adversary \mathcal{A} , it holds that

$$|\Pr[\mathcal{A}(\mathsf{PRG}(U')) = 1] - \Pr[\mathcal{A}(U) = 1]| = \mathsf{negl}(\lambda),$$

where U and U' are uniform distributions over $\{0,1\}^m$ and $\{0,1\}^{\lambda}$, respectively.

Observe that, given any pseudorandom generator PRG, one could expand it into a new PRG' of an arbitrarily-large polynomial stretch (i.e., m/λ) by repeatedly applying PRG.

Definition 5 (PRF). A function $\mathsf{PRF} : \{0,1\}^{\lambda} \times \{0,1\}^{\mathsf{poly}(\lambda)} \to \{0,1\}^{\mathsf{poly}(\lambda)}$ is called a pseudorandom function if for all PPT adversary \mathcal{A} , it holds that

$$\left|\Pr\left[\mathcal{A}^{\mathsf{PRF}(k,\cdot)}(1^{\lambda})=1\right]-\Pr\left[\mathcal{A}^{F(\cdot)}(1^{\lambda})=1\right]\right|=\mathsf{negl}(\lambda),$$

where k is sampled uniformly at random and function F is sampled uniformly at random from the set of all functions.

Definition 6 (SKE). A symmetric-key encryption (SKE) scheme consists of three algorithms (Gen, Enc, Dec) with the following syntax.

- $\mathsf{sk} \leftarrow \mathsf{Gen}(1^{\lambda})$: on input the security parameter, Gen outputs a secret key sk .
- ct \leftarrow Enc(sk, msg): on input the secret key sk and a message msg, Enc outputs a ciphertext ct.
- msg = Dec(sk, ct): on input the secret key sk and a ciphertext ct, Dec outputs a message msg.

It satisfies the following guarantee:

- Correctness. For all message msg, it holds that Pr[Dec(sk, ct) = msg] = 1, where sk ← Gen(1^λ) and ct ← Enc(sk, msg).
- **CPA-security.** For all messages msg_0, msg_1 and PPT adversary \mathcal{A} , it holds that

$$\left| \Pr \begin{bmatrix} \mathsf{sk} \leftarrow \mathsf{Gen}(1^{\lambda}), \ b \leftarrow \{0, 1\}, \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathsf{msg}_b), \ b' \leftarrow \mathcal{A}^{\mathsf{Enc}(\mathsf{sk}, \cdot)}(\mathsf{ct}) \\ \end{array} \right| = \mathsf{negl}(\lambda).$$

3.4 Garbled Circuits

Definition 7 (GC). A garbling scheme consists of two algorithms (Garble, Eval). The garbling algorithm takes a circuit C as input and outputs a garbled circuit \hat{C} and a label for each input wire and value, i.e., $(\hat{C}, \{\mathsf{lab}_{w,b}\}_{w,b}) \leftarrow \mathsf{Garble}(1^{\lambda}, C)$. The evaluation algorithm takes as input a garbled circuit and one label for each input wire and outputs an evaluation, i.e., $y = \mathsf{Eval}(\hat{C}, \{\mathsf{lab}_{w,x_w}\}_w)$. It satisfies the following guarantee.

• Correctness. For any circuit C and input x, it holds that

$$\Pr\begin{bmatrix} \left(\widehat{C}, \{\mathsf{lab}_{w,b}\}_{w,b}\right) \leftarrow \mathsf{Garble}(1^{\lambda}, C) \\ y = \mathsf{Eval}\left(\widehat{C}, \{\mathsf{lab}_{w,x_w}\}_w\right) &: C(x) = y \end{bmatrix} = 1.$$

• Security. There exists a simulator Sim such that for any circuit C and input x, it holds that

$$\left(\widehat{C}, \{\mathsf{lab}_{w,x_w}\}_w\right) \stackrel{c}{\approx} \mathsf{Sim}\left(1^{\lambda}, 1^{|C|}, C(x)\right),$$

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

3.5 Extractable Commitment

Definition 8. A three-round extractable commitment consists of three algorithms ($\mathsf{Ecom}_1, \mathsf{Ecom}_2, \mathsf{Ecom}_3$). In the first round, the committer with message x, sends $\mathsf{ecom}_1 = \mathsf{Ecom}_1(x;\tau)$. The receiver is public-coin, which means that $\mathsf{ecom}_2 \leftarrow \mathsf{Ecom}_2$ is simply a uniformly random string. Given ecom_2 , in the third round, the committer sends $\mathsf{ecom}_3 = \mathsf{Ecom}_3(x,\tau,\mathsf{ecom}_2)$. The scheme is statistically binding in that, given the first message $ecom_1$, it is statistically impossible for the committer to open to a different message x'. We also require the existence of an extractor EcomExt such that for any malicious committer C^* , the following holds.

$$\Pr\begin{bmatrix} (\mathsf{ecom}_1, \mathsf{ecom}_2, \mathsf{ecom}_3) \leftarrow \langle \mathcal{C}^*, \mathcal{R} \rangle \text{ is accepting and} \\ x \leftarrow \mathsf{EcomExt}^{\mathcal{C}^*}(\mathsf{ecom}_1, \mathsf{ecom}_2, \mathsf{ecom}_3) \text{ such that} \\ \exists x', \tau \text{ s.t. } (x' \neq x) \land \mathsf{ecom}_1 = \mathsf{Ecom}_1(x'; \tau) \end{bmatrix} = \mathsf{negl}(\lambda).$$

In particular, the extractor EcomExt requires two accepting transcripts (i.e., $ecom_1$, $\{ecom_2^j, ecom_3^j\}_{j=1}^2$), such that $ecom_2^1$ and $ecom_2^2$ are distinct, to extract a message x.

Such a scheme can be constructed based on injective one-way functions (see, for example, [PW09]).

Delayed-input Extractable Commitment. We could use the above extractable commitment scheme to construct a delayed-input extractable commitment. That is, the committer only obtains the message m to be committed after the first two rounds. Intuitively, the committer simply commits to a random string k using the above scheme, and, in the last round, it also sends the ciphertext $m \oplus k$.⁴ Moreover, one could still use the extractor to extract m as extracting k immediately reveals m.

Formally, this scheme is as follows.

- 1. In the first round, the committer samples a random string k. Compute and send $ecom_1 = Ecom(k; \tau)$.
- 2. In the second round, the receiver samples and sends $ecom_2 \leftarrow Ecom_2$.
- 3. To commit to a message m, the committer computes and sends $(m \oplus k, \text{ ecom}_3 = \text{Ecom}_3(k, \tau, \text{ecom}_2))$.

Observe that the hiding property holds even in the following sense. Consider the game where the adversary sends two messages m_0 and m_1 to the challenger and engages in an arbitrary polynomial number of independent instances of the extractable commitment scheme with the challenger. The challenger samples a random bit b and commits to m_b in all instances. Even in this case, the adversary cannot predict b with non-negligible advantage. One can prove this using a standard hybrid argument.

To differ from the non-delayed-input commitment scheme, we shall explicitly state that the scheme is delayed-input whenever we use it.

Delayed-input Reusable Extractable Commitment. Similarly, we could use the extractable commitment scheme to construct a delay-input *reusable* extractable commitment. That is, the first two rounds are executed only once. Afterward, the committer could send multiple third-round messages to commit to different messages. Intuitively, one commits to a secret-key sk for SKE. In every reuse session, the sender appends the encryption of the message under sk along with the original third round message of the extractable commitment scheme.

Formally, this scheme is as follows.

- 1. In the first round, the committer samples a secret key sk for SKE. Compute and send $ecom_1 = Ecom(sk; \tau)$.
- 2. In the second round, the receiver samples and sends $ecom_2 \leftarrow Ecom_2$.
- 3. To commits to a message m in a reuse session, the committer computes and sends $(\mathsf{Enc}(\mathsf{sk}, m), \mathsf{ecom}_3 =$

 $\mathsf{Ecom}_3(\mathsf{sk}, \tau, \mathsf{ecom}_2)$ where Enc is a public-coin symmetric key encryption.⁵

Observe that this reusable extractable commitment is hiding due to the hiding property of the extractable commitment scheme and the semantic security of the symmetric key encryption scheme. The binding property of the construction follows directly from the binding of the underlying extractable commitment. Furthermore, extracting the key k enables the extraction of all the messages m. We shall use (rEcom₁, rEcom₂, rEcom₃) to denote this reusable extractable commitment scheme.

⁴Observe that the delayed-input scheme is no longer first-message binding.

⁵Note that standard SKE scheme based on PRF where the ciphertext comprises of $(r, \mathsf{PRF}(\mathsf{sk}, r) \oplus m)$ (where r is uniformly chosen) is a public-coin SKE construction.

3.6 ZAP

ZAP [DN00] is a two-message public-coin witness-indistinguishable proof.

Definition 9. A ZAP for a language $\mathcal{L} \in \mathsf{NP}$ with relation $\mathcal{R}_{\mathcal{L}}$ consists of two algorithms $(\mathcal{P}, \mathcal{V})$ such that the following hold.

• Completeness. There exists a polynomial $p(\lambda)$ such that for every $(x, w) \in \mathcal{R}_{\mathcal{L}}$,

$$\Pr\left[r \leftarrow \{0,1\}^{p(\lambda)}, \ \pi \leftarrow \mathcal{P}(x,w,r) : \ \mathcal{V}(x,r,\pi) = 1\right] = 1.$$

• Soundness. For any PPT adversary \mathcal{P}^* , it holds that

$$\Pr\left[r \leftarrow \{0,1\}^{p(\lambda)}, \ (x,\pi) \leftarrow \mathcal{P}^*(r) : \ \left(x \notin \mathcal{L}\right) \land \left(\mathcal{V}(x,r,\pi) = 1\right)\right] = \mathsf{negl}(\lambda).$$

 Witness Indistinguishability. For any statement x and witnesses w₁ and w₂ such that (x, w₁) ∈ R_L and (x, w₂) ∈ R_L and for any PPT adversary V^{*}, it holds that

$$\begin{split} |\Pr[r \leftarrow \mathcal{V}^*, \ \pi \leftarrow \mathcal{P}(x, w_1, r) : \ \mathcal{V}^*(r, x, \pi) = 1] - \\ & \Pr[r \leftarrow \mathcal{V}^*, \ \pi \leftarrow \mathcal{P}(x, w_2, r) : \ \mathcal{V}^*(r, x, \pi) = 1]| = \mathsf{negl}(\lambda). \end{split}$$

Note that ZAP is immediately reusable. That is, given a fixed first-round message by the verifier, the witness indistinguishability is preserved even if the prover sends multiple second-round messages proving (possibly different) statements using different witnesses.

For ease of presentation, in our protocols, we shall use (ZAP_1, ZAP_2) to denote the algorithms that generate the first and second message of a ZAP protocol.

3.7 Trapdoor Generation Protocol

A trapdoor generation protocol is a protocol between the sender and receiver. The sender does not hold any input and the receiver shall not output anything. The objective of the trapdoor generation protocol is for the sender to establish a trapdoor. The security is two-fold. First, a malicious receiver shall not be able to generate a valid trapdoor. Second, there is an extractor who could extract a valid trapdoor by rewinding the sender. Similar to [BGJ⁺18, CCG⁺20], we define trapdoor generation protocol as follows.

A three-round trapdoor generation protocol consists of the following algorithms (TDGen₁, TDGen₂, TDGen₃, TDOut, TDValid, TDExt). In the first round, the sender sends $td_1 \leftarrow TDGen_1(r_S)$. In the second round, the receiver sends $td_2 \leftarrow TDGen_2$. In the third round, the sender sends $td_3 = TDGen_3(r_S, td_2)$. The trapdoor generation succeeds if $TDOut(td_1, td_2, td_3) = 1$. Given any input t and the first round message td_1 , $TDValid(t, td_1)$ checks whether t is a valid trapdoor or not. It satisfies the following security guarantee.

- A malicious PPT receiver could not generate a valid trapdoor.
- There is an extractor who outputs a valid trapdoor via rewinding. In particular, given two accepting transcripts $(\mathsf{td}_1, \{\mathsf{td}_2^i, \mathsf{td}_3^i\}_{i=1}^2)$ such that td_2^1 and td_2^2 are distinct, TDExt outputs a valid trapdoor t.

Badrinarayanan et al. [BGJ⁺18] gave a construction of such a trapdoor generation protocol based on digital signatures which can in turn be based on one-way functions [Rom90].

1-rewind Security. A stronger notion called 1-rewind security is needed for the construction of our MPC protocol. Consider the following experiment between the honest sender S and a malicious PPT receiver \mathcal{R}^* .

- The honest sender and malicious receiver engage in the trapdoor generation protocol once. The transcript is (td_1, td_2, td_3)
- Then, the malicious receiver is allowed to rewind back to the beginning of round 2 and sends another second-round message td'_2 . It receives back td'_3 as the third round message.

• The malicious receiver outputs a string t based on its view.

The trapdoor generation protocol is 1-rewind secure if for any malicious receiver \mathcal{R}^* , in the above experiment, it holds that

$$\Pr[\mathsf{TDValid}(t, \mathsf{td}_1) = 1] = \mathsf{negl}(\lambda).$$

For the 1-rewind secure trapdoor generation protocol, the extractor requires three accepting transcripts to extract a trapdoor. That is, $(\mathsf{td}_1, \{\mathsf{td}_2^i, \mathsf{td}_3^i\}_{i=1}^3)$ such that $\mathsf{td}_2^1, \mathsf{td}_2^2, \mathsf{td}_2^3$ are all distinct.

A 1-rewind secure trapdoor generation protocol can be constructed from any signature scheme, which can be based on one-way functions. We refer the readers to, for example, $[CCG^+20]$ for such a construction.

4 Reusable Zero-knowledge Argument of Knowledge

In this section, we give a construction of a round-optimal reusable zero-knowledge argument of knowledge (ZK-AoK) protocol where the first three rounds of the protocol could be reused an unbounded number of times to prove different statements. Specifically, the prover and the verifier execute the first three rounds of the protocol once. Afterward, given any language \mathcal{L} in NP and $(st, w) \in \mathcal{R}_{\mathcal{L}}$, the prover could send the fourth round of the protocol to convince the verifier that the statement st is in the language \mathcal{L} . The fourth round could be re-executed for an unbounded number of times. We require the proof to be zero-knowledge and, additionally, the existence of a knowledge extractor. We start with the syntax and give the formal definition.

Syntax. A four-round reusable zero-knowledge argument of knowledge protocol consists of a tuple of algorithms ($rZK_1, rZK_2, rZK_3, rZK_4, out_{\mathcal{V}}$). To generate the messages of the protocol in the first three rounds, the Prover \mathcal{P} uses randomness $r_{\mathcal{P}}$ and the verifier uses randomness $r_{\mathcal{V}}$. That is, the verifier sends $rzk_1 = rZK_1(1^{\lambda}; r_{\mathcal{V}})$; the prover sends $rzk_2 = rZK_2(1^{\lambda}, rzk_1; r_{\mathcal{P}})$; the verifier sends $rzk_3 = rZK_3(1^{\lambda}, rzk_2; r_{\mathcal{V}})$.

 $\mathsf{rZK}_1(1^{\lambda}; r_{\mathcal{V}}); \text{ the prover sends } \mathsf{rzk}_2 = \mathsf{rZK}_2(1^{\lambda}, \mathsf{rzk}_1; r_{\mathcal{P}}); \text{ the verifier sends } \mathsf{rzk}_3 = \mathsf{rZK}_3(1^{\lambda}, \mathsf{rzk}_2; r_{\mathcal{V}}).$ In the *i*th reuse session, the prover with input $(\mathsf{st}^{(i)}, w^{(i)}) \in \mathcal{R}_{\mathcal{L}^{(i)}}$ sends the proof $\pi^{(i)} \leftarrow \mathsf{rZK}_4(\mathsf{rzk}_1, \mathsf{rzk}_3, r_{\mathcal{P}}, \mathsf{st}^{(i)}, w^{(i)}).^6$ The verifier with input $\mathsf{st}^{(i)}$ verifies the validity of the proof using $\mathsf{out}_{\mathcal{V}}(\mathsf{rZK}_1, \mathsf{rzk}_2, \mathsf{rzk}_3, \mathsf{st}^{(i)}, \pi^{(i)}).^7$

Definition 10. A four-round protocol rZK between a prover and a verifier is a reusable zero-knowledge argument of knowledge if it satisfies the following.

- Completeness. In honest execution, the verifier always accepts the proof.
- Zero-knowledge. For any malicious stateful PPT verifier \mathcal{V}^* , there exists a stateful (expected) PPT simulator Sim such that, for all $\ell = \text{poly}(\lambda)$ and for any non-uniform (stateful) PPT distinguisher D, the real and the ideal executions described below are computationally indistinguishable.
 - Real Execution.
 - * The honest prover and the verifier \mathcal{V}^* (which is given $1^{\lambda}, 1^{\ell}$) execute the first three rounds of the protocol. Let $r_{\mathcal{P}}$ denote the random tape used by the prover to generate the first three rounds and let $(\mathsf{rzk}_1, \mathsf{rzk}_2, \mathsf{rzk}_3)$ be the transcript.
 - * For the *i*-th reuse session:
 - The distinguisher D on input the current view of \mathcal{V}^* outputs $(\mathsf{st}^{(i)}, w^{(i)})$. If $(\mathsf{st}^{(i)}, w^{(i)}) \in \mathcal{R}_{\mathcal{L}^{(i)}}$, then the prover generates $\pi^{(i)} \leftarrow \mathsf{rZK}_4(\mathsf{rzk}_1, \mathsf{rzk}_3, r_{\mathcal{P}}, \mathsf{st}^{(i)}, w^{(i)})$ and otherwise, it sets $\pi^{(i)} = \bot$. It sends $\pi^{(i)}$ to the verifier \mathcal{V}^* .
 - * The output of the real execution corresponds to the final output of D that is given as input the view of \mathcal{V}^* at the end of the ℓ -th reuse session.
 - Ideal Execution.
 - * The simulator Sim on input $(1^{\lambda}, 1^{\ell})$ and oracle access to \mathcal{V}^* generates the view of \mathcal{V}^* at the end of the third round and outputs it.

 $^{{}^{6}}r_{\mathcal{P}}$ is only the secret state of the prover. The prover has access to fresh randomness.

⁷Note that the verifier does not hold any secret state. Hence, given the first three rounds, the proof is publicly verifiable.

- * The ideal execution initializes \mathcal{V}^* with this view and continues with the rest of the execution as follows.
- * In the *i*-th reuse session:
 - The distinguisher D on input the current view of \mathcal{V}^* outputs $(\mathsf{st}^{(i)}, w^{(i)})$.
 - · If $(\mathsf{st}^{(i)}, w^{(i)}) \in \mathcal{R}_{\mathcal{L}^{(i)}}$, Sim on input $\mathsf{st}^{(i)}$ generates $\pi^{(i)}$. Else, we set $\pi^{(i)} = \bot$. We send $\pi^{(i)}$ to the verifier \mathcal{V}^* .
- * The output of the ideal execution corresponds to the final output of D that is given as input the view of \mathcal{V}^* at the end of the ℓ -th reuse session.
- *Knowledge Extraction.* For any malicious prover \mathcal{P}^* , there exists an (expected) PPT knowledge extractor E such that, for all statement $\mathsf{st}^{(i)}$, it holds that

$$\Pr\left[\left(\mathsf{st}^{(i)}, \, \mathsf{E}^{\mathcal{P}^*}(\mathsf{st}^{(i)})\right) \in \mathcal{R}_{\mathcal{L}^{(i)}}\right] \ge \Pr\left[\mathsf{Accept}^{(i)}\right] - \mathsf{negl}(\lambda),$$

if $\Pr\left[\mathsf{Accept}^{(i)}\right]$ is non-negligible where $\mathsf{Accept}^{(i)}$ denotes the event that the honest verifier accepts the proof in the *i*th reuse session when interacting with the malicious prover \mathcal{P}^* .

4.1 Construction

In Figure 1, we present our construction with the following building blocks.

- A delayed-input reusable extractable commitment scheme $(\mathsf{rEcom}_1, \mathsf{rEcom}_2, \mathsf{rEcom}_3)$ from Section 3.5.
- A delayed-input extractable commitment scheme (Ecom₁, Ecom₂, Ecom₃) from Section 3.5.
- A trapdoor generation protocol (TDGen₁, TDGen₂, TDGen₃, TDOut, TDValid) from Section 3.7.
- A ZAP scheme (ZAP_1, ZAP_2) from Section 3.6.
 - Language $\widehat{\mathcal{L}^{(i)}}$. Fix the transcript for the first three rounds. A statement

$$\widehat{\mathsf{st}^{(i)}} = \left((\mathsf{recom}_1, \mathsf{recom}_2, \mathsf{recom}_3^{(i)}), (\mathsf{ecom}_1, \mathsf{ecom}_2, \mathsf{ecom}_3), \mathsf{st}^{(i)} \right)$$

is in language $\widehat{\mathcal{L}^{(i)}}$ with witness $(w^{(i)}, r_{\mathsf{recom}}, t, r_{\mathsf{ecom}})$ if either one of the following holds.

- Valid Witness. (recom₁, recom₂, recom₃⁽ⁱ⁾) is an honest commitment of $w^{(i)}$ with randomness r_{recom} and $(\mathsf{st}^{(i)}, w^{(i)}) \in \mathcal{R}_{\mathcal{L}^{(i)}}$.
- Valid Trapdoor. TDValid $(t, td_1) = 1$ and $(ecom_1, ecom_2, ecom_3)$ is an honest commitment of t with randomness r_{ecom} .
- Protocol description.
 - 1. Round-1: The verifier sends $\mathsf{td}_1 \leftarrow \mathsf{TDGen}_1(r_{\mathsf{td}})$.
 - 2. Round-2: The prover samples the following $\mathsf{td}_2 \leftarrow \mathsf{TDGen}_2$, recom₁ $\leftarrow \mathsf{rEcom}_1(r_{\mathsf{recom}})$, and $\mathsf{ecom}_1 \leftarrow \mathsf{Ecom}_1(r_{\mathsf{ecom}})$. The prover sends td_2 , recom_1 , and $\underline{\mathsf{ecom}_1}$.
 - 3. Round-3: The verifier samples the following $\mathsf{td}_3 = \mathsf{TDGen}_3(r_{\mathsf{td}}, \mathsf{td}_2)$, $\mathsf{recom}_2 \leftarrow \mathsf{FEcom}_2$, $\mathsf{ecom}_2 \leftarrow \mathsf{Ecom}_2$, and $\mathsf{zap}_1 \leftarrow \mathsf{ZAP}_1$. The verifier sends td_3 , recom_2 , ecom_2 , $\mathsf{and} \mathsf{zap}_1$.
 - 4. Round-4: For the i^{th} reuse session, prover does the following.
 - The prover checks that the trapdoor generation is successful, i.e., $\mathsf{TDOut}(\mathsf{td}_1, \mathsf{td}_2, \mathsf{td}_3) = 1$. If not, the prover aborts. Otherwise, the prover continues to sample: $\mathsf{recom}_3^{(i)} = \mathsf{rEcom}_3(\mathsf{recom}_2, r_{\mathsf{recom}}, w^{(i)}),$ $\mathsf{ecom}_3 = \mathsf{Ecom}_3(\mathsf{ecom}_2, r_{\mathsf{ecom}}, 0^{\lambda}),$

 $\operatorname{zap}_{2}^{(i)} \leftarrow \operatorname{ZAP}_{2}(\operatorname{zap}_{1}, \widehat{\operatorname{st}^{(i)}}, (w^{(i)}, r_{\operatorname{recom}}, \bot, \bot))$ proving that $\widehat{\operatorname{st}^{(i)}} = ((\operatorname{recom}_{1}, \operatorname{recom}_{2}, \operatorname{recom}_{3}^{(i)})$ (ecom₁, ecom₂, ecom₃), st⁽ⁱ⁾) $\in \widehat{\mathcal{L}^{(i)}}$ with witness $(w^{(i)}, r_{\operatorname{recom}}, \bot, \bot)$. The prover sends recom₃⁽ⁱ⁾, ecom₃, and $\operatorname{zap}_{2}^{(i)}$.

5. Verifier Output. The verifier accepts the proof if both $(\mathsf{recom}_1,\mathsf{recom}_2,\mathsf{recom}_3^{(i)})$ and $(\mathsf{ecom}_1,\mathsf{ecom}_2,\mathsf{ecom}_3)$ are accepting transcripts, and $(\mathsf{zap}_1,\mathsf{zap}_2)$ is a valid proof of the statement $\widehat{\mathsf{st}^{(i)}}$.

Figure 1: Our four-round reusable ZK-AoK

We shall prove the following theorem. The completeness follows immediately.

Theorem 3. Assuming the security of the reusable extractable commitment scheme, the extractable commitment scheme, the trapdoor generation protocol, and the ZAP, the protocol described in Figure 1 is a reusable ZK-AoK (see Definition 10).

4.2 Proof of Zero-knowledge

The simulator is presented in Figure 2. We shall prove the indistinguishability through the following hybrids, where H_4 is identical to the simulator.

• Extracting the Trapdoor.

- The simulator runs the first three rounds of the protocol using the same strategy as that of the honest prover. If the trapdoor generation fails, i.e., $\mathsf{TDOut}(\mathsf{td}_1, \mathsf{td}_2, \mathsf{td}_3) \neq 1$, the simulator aborts and outputs the view of the adversary.
- The simulator fixes the first round of the protocol. It generates a number of lookahead threads for the second and third rounds of the protocol. It generates as many as needed until it gets two accepting transcripts. That is, there are td₁, {td^j₂, td^j₃}²_{j=1} such that TDOut(td₁, td^j₂, td^j₃) = 1. If td¹₂ and td²₂ are not distinct, the simulator aborts. Otherwise, it invokes the TDExt(td₁, {td^j₂, td^j₃}²_{j=1}) to extract a valid trapdoor t*.
- Commit to the trapdoor and prove using the trapdoor.
 - The simulator generates independently sampled second-round messages and sends them to the verifier until the verifier replies back a valid third-round message (i.e., the trapdoor generation does not fail). The simulator outputs the view of the adversary in the first three rounds.
 - To generate the final round message in the *i*-th reuse session, the simulator on input $st^{(i)}$ generates the fourth round as follows. It computes

 $\operatorname{\mathsf{recom}}_{3}^{(i)} = \operatorname{\mathsf{rEcom}}_{3}(\operatorname{\mathsf{recom}}_{2}, r_{\operatorname{\mathsf{recom}}}, r^{(i)}) \text{ where } r^{(i)} \text{ is a random string, } \operatorname{\mathsf{ecom}}_{3} = \operatorname{\mathsf{Ecom}}_{3}(\operatorname{\mathsf{ecom}}_{2}, r_{\operatorname{\mathsf{ecom}}}, t^{*}), \operatorname{\mathsf{zap}}_{2}^{(i)} \leftarrow \operatorname{\mathsf{ZAP}}_{2}(\operatorname{\mathsf{zap}}_{1}, \operatorname{st}^{(i)}, (\bot, \bot, t^{*}, r_{\operatorname{\mathsf{ecom}}})) \text{ proving that } \operatorname{st}^{(i)} \in \widehat{\mathcal{L}}^{(i)} \text{ with witness } (\bot, \bot, t^{*}, r_{\operatorname{\mathsf{ecom}}}). \text{ The simulator sends } \operatorname{\mathsf{recom}}_{3}^{(i)}, \operatorname{\mathsf{ecom}}_{3}, \operatorname{and} \operatorname{\mathsf{zap}}_{2}^{(i)} \text{ to the verifier.}$

Figure 2: The simulator for zero-knowledge

Running Time. Let ε be the probability that the malicious verifier sends a valid third-round message. Then, the expected running time of the above simulator is given by $(1 - \varepsilon)\mathsf{poly}(\lambda) + \varepsilon \cdot (\frac{\mathsf{poly}(\lambda)}{\varepsilon}) = \mathsf{poly}(\lambda)$.

Hybrid description.

• H_0 : This is the real execution.

- H_1 : In this hybrid, if the trapdoor generation succeeds, it freezes the first round and generates a number of lookahead threads for the second and third rounds. It generates as many as needed until it gets two accepting transcripts. The simulator aborts if a valid trapdoor t^* is not extracted.
- H_2 : In the fourth round, the simulator now commits to the valid trapdoor t^* using the delayed-input extractable commitment in the main thread, i.e., $ecom_3 = Ecom_3(ecom_2, r_{ecom}, t^*)$ (instead of committing to all zeroes string).
- H₃: In the fourth round, the simulator now proves the statement st⁽ⁱ⁾ using the witness (⊥, ⊥, t^{*}, r_{ecom}) (instead of the witness (w⁽ⁱ⁾, r_{recom}, ⊥, ⊥)).
- H_4 : In the fourth round, the simulator now commits to a random string $r^{(i)}$ (instead of the witness $w^{(i)}$) in the reusable extractable commitment, i.e., $\operatorname{recom}_3^{(i)} = \operatorname{rEcom}_3(\operatorname{recom}_2, r_{\operatorname{recom}}, r^{(i)})$.

Security Proof.

Claim 1. H_0 and H_1 are indistinguishable assuming the trapdoor extraction property of the trapdoor generation protocol.

Proof. The only difference between the two hybrids is when the trapdoor extraction fails. We shall prove that the running time of the simulator is expected polynomial time and the failure probability is exponentially low.

Suppose the probability that the simulator receives an accepting third round message is ε . Therefore, with probability $1 - \varepsilon$, the third round message is not accepting and the simulator aborts. Otherwise, the simulator will try to rewind to generate another accepting transcript. In expectation, it needs to generate $1/\varepsilon$ lookahead threads to obtain an accepting transcript. Therefore, in expectation, the simulator will generate $(1 - \varepsilon) + \varepsilon \cdot \frac{1}{\varepsilon}$ number of threads. Hence, the simulator is efficient.

Since $\mathsf{td}_2 \leftarrow \mathsf{TDGen}_2$ is a random string of length λ , the probability that $\mathsf{td}_2^1 = \mathsf{td}_2^2$ is $\exp(-\Theta(\lambda))$. Therefore, the failure probability is upper-bounded by $\varepsilon \cdot \frac{1}{\varepsilon} \cdot \exp(-\Theta(\lambda))$, which is exponentially low. \Box

Claim 2. H_1 and H_2 are indistinguishable assuming the hiding property of the delayed-input extractable commitment scheme.

Proof. If the two hybrids are not indistinguishable, we could use it to construct an adversary \mathcal{A}_{ecom} that breaks the hiding property of the delayed-input extractable commitment scheme.

The \mathcal{A}_{ecom} interacts with the malicious verifier in the exact same way as in H_1 . That is, it freezes the first message and extracts a valid trapdoor t^* as in H_1 . It forwards t^* and 0^{λ} to the challenger \mathcal{C}_{ecom} . It then invokes the challenger multiple times to generate independently sampled $ecom_1$ and uses it to generate the second round message in the main thread until it obtains a valid third round message from the verifier. It forwards the corresponding $ecom_2$ to the challenger and receives $ecom_3$. It uses this to complete the fourth round message in the protocol. Depending on whether the challenger commits to t^* or 0^{λ} , it either simulates the hybrid H_1 or H_2 . Furthermore, the simulator in this reduction is efficient as its expected running time is given by $(1 - \varepsilon)\operatorname{poly}(\lambda) + \varepsilon \cdot (\frac{\operatorname{poly}(\lambda)}{\varepsilon}) = \operatorname{poly}(\lambda)$ where ε is the probability that verifier outputs a valid third round message. Since the two hybrids are distinguishable, \mathcal{A}_{ecom} breaks the hiding property of the commitment scheme.

Claim 3. H_2 and H_3 are indistinguishable assuming the witness indistinguishable property of the ZAP.

Proof. If the two hybrids are not indistinguishable, we shall break the witness indistinguishable property of the ZAP.

The \mathcal{A}_{zap} interacts with the malicious verifier in the exact same way as the hybrids. That is, it freezes the first message and extracts a valid trapdoor t^* . Since the two hybrids are distinguishable, there is a non-negligible probability that this happens. \mathcal{A}_{zap} then simulates the second round message in the main thread as in H_2 and obtains a valid third round message. It forwards zap_1 from this third round message to the external challenger. For the *i*-th reuse session, \mathcal{A}_{zap} forwards the statement $\widehat{st^{(i)}}$ and the two witnesses (i.e., $(\perp, \perp, t^*, r_{ecom})$ and $(w^{(i)}, r_{recom}, \perp, \perp))$ to the challenger \mathcal{C}_{zap} . It forwards the zap_2 it receives from the challenger as $z_{ap}^{(i)}$. Depending on which witness the challenger uses, it either simulates H_2 or H_3 . Since the two hybrids are distinguishable, \mathcal{A}_{zap} break the witness indistinguishable property of the ZAP.

Claim 4. H_3 and H_4 are indistinguishable assuming the hiding property of the reusable extractable commitment scheme.

Proof. The proof is similar to Claim 2.

4.3 **Proof of Knowledge Extraction**

The knowledge extractor is presented in Figure 3. Let μ be the probability that prover sends an accepting transcript. By assumption μ is non-negligible. To prove the knowledge extraction property, we prove the following claims.

- Check validity of the proof. The knowledge extractor runs full execution of the protocol as the honest verifier and checks if the honest verifier accepts the proof or not.
- Rewind the third round to extract the witness. If the proof is accepting, the knowledge extractor rewinds the third round as many times as needed to obtain two accepting transcript recom₁, $\{\text{recom}_2^j, \text{recom}_3^j\}_{j=1}^2$. If recom_2^1 and recom_2^2 are not distinct, the knowledge extractor aborts. Otherwise, it extracts the key k present in the extractable commitment and uses this key to extract message $w^{(i)}$ sent in the last round (recall that in our reusable delayed-input extractable commitment, a key k is committed using a standard extractable commitment and the message to be committed is encrypted under this key k and sent in the last round.). It outputs $w^{(i)}$ as the witness.

Figure 3: The knowledge extractor

Claim 5. Let $Event_1$ be the event that there exists a valid trapdoor t^* and randomness r^*_{ecom} such that the transcript ($ecom_1, ecom_2, ecom_3$) in the main thread is a commitment to t^* with randomness r^*_{ecom} . Assuming the security of the trapdoor generation protocol, it holds that

$$\Pr[\mathsf{Event}_1] = \mathsf{negl}(\lambda).$$

Proof. Suppose this event happens with non-negligible probability, we shall break the security of the trapdoor generation protocol as follows.

 \mathcal{A}_{td} receives the td₁ from the challenger \mathcal{C}_{td} . It forwards this message to the malicious prover. It sends the td₂ it receives from the prover to the challenger from \mathcal{C}_{td} and receives back td₃. It sends td₃ in the third message to the malicious prover. It rewinds the third message as many times as needed to obtain two accepting transcripts $ecom_1, \{ecom_2^j, ecom_3^j\}_{j=1}^2$. Based on a similar argument as in Claim 1, this step runs in expected polynomial time and the failure probability (i.e., $ecom_2^1 = ecom_2^2$) is negligible. It extracts the trapdoor t that is committed. The extractor of the commitment scheme guarantees that the committed value cannot be anything other than t. The fact that Event₁ happens with non-negligible probability implies that t is a valid trapdoor with non-negligible probability (follows from the correctness of extraction of ecom). Hence, \mathcal{A}_{td} outputs a valid trapdoor t with non-negligible probability, which breaks the security of the trapdoor generation protocol.

Claim 6. Let Event_2 be the event that there exists a valid witness $w^{(i)}$ and randomness r^*_{recom} such that the transcript $(\text{recom}_1, \text{recom}_2, \text{recom}_3^{(i)})$ in the main thread is a commitment to a valid witness $w^{(i)}$ with randomness r^*_{recom} . Assuming the soundness of the ZAP,

$$\Pr[\mathsf{Event}_2] \ge 1 - \mathsf{negl}(\lambda)$$

Proof. We first show that if zap_2 is accepting then it follows from the soundness of ZAP that $\hat{st}^{(i)}$ in the language with probability $1 - negl(\lambda)$. Assume for the sake of contradiction that this is not the case and we now give a reduction that breaks the soundness of ZAP.

 \mathcal{A}_{zap} receives the first message r from the challenger \mathcal{C}_{zap} . It simulates the main thread of the protocol with r as zap_1 . The prover sends a valid proof zap_2 proving the statement $\widehat{st}^{(i)}$. We forward this to the ZAP challenger. Since zap_2 is accepting with non-negligible probability μ , it follows that the probability that the statement $\widehat{st}^{(i)}$ is not in the language is negligible.

Since $\widehat{st}^{(i)}$ is in the language with overwhelming probability and by Claim 5, $(ecom_1, ecom_2, ecom_3)$ is not a valid commitment to the trapdoor with overwhelming probability, it follows that Event₂ happens with overwhelming probability.

From Claims 5, 6, it follows that the probability that $\mathsf{Event}_1 \lor \neg \mathsf{Event}_2$ is $\mathsf{negl}(\lambda)$, it follows from the correctness of extraction of recom that the output of the knowledge extractor is a valid witness with probability at least $\mu - \mathsf{negl}(\lambda)$.

5 Reusable Oblivious Transfer Protocol

In this section, we give a construction of a reusable four-round oblivious transfer protocol. That is, the sender and receiver shall only execute the first two rounds once. Afterward, given a new choice bit as input, the receiver could send a new third-round message. As for the sender, on fixing the first three rounds, the sender could send multiple fourth-round messages given different pairs of messages as inputs.

We first make the following observation on how to transform any OT protocol into one where the first three rounds of protocol could be reused an unbounded number of times.

Remark 2 (Any OT is sender reusable). Given any four-round OT protocol, one could transform it into a new four-round OT protocol such that the first three rounds are reusable. Specifically, the sender could execute the original OT protocol with two keys k_0 and k_1 as his inputs. To reuse the first three rounds with different sender's input m_0 and m_1 , the sender could simply append the fourth round message with a public-coin symmetric key encryption of m_0 and m_1 using k_0 and k_1 respectively. The sender's privacy for the original OT protocol implies its privacy for the reusable OT protocol. Furthermore, this transformation preserves receiver reusability.

In light of this observation, we shall only focus on the reusability of the first two rounds where the receiver's choice bit may change.

As a first step, we shall construct a reusable oblivious transfer protocol that provides indistinguishability security against a malicious sender and simulation security against a malicious receiver, which we define below. In the next section, we use this protocol as our main building block and construct a reusable 2PC protocol.

Syntax. A four-round reusable OT protocol consists of the following algorithms $(rOT_1, rOT_2, rOT_3, rOT_4, out_{rOT})$. Let r_S and r_R denote the sender's and receiver's private randomness in the first two rounds. The receiver sends $ot_1 = rOT_1(1^{\lambda}; r_R)$ in the first round. The sender sends $ot_2 \leftarrow rOT_2(1^{\lambda}, ot_1; r_S)$ in the second round. For the i^{th} reuse session, let us denote the receiver's choice bit by $b^{(i)}$ and the sender's messages by $m_0^{(i)}, m_1^{(i)}$. Receiver shall send $ot_3^{(i)} = rOT_3(1^{\lambda}, ot_2, r_R, b^{(i)})$ as the third round message. The sender shall send $ot_4^{(i)} = rOT_4(1^{\lambda}, ot_1, ot_3^{(i)}, m_1^{(i)}, r_S)$ as the fourth round message.⁸ The receiver finally runs out_{rOT} on the transcript of the protocol and its entire random tape to compute the output.

Definition 11. We say that rOT is a four-round reusable oblivious transfer protocol with simulation security against corrupted receivers and indistinguishability security against corrupted senders if:

- Correctness. For all *i*, the receiver's output for the *i*th session is $m_{h^{(i)}}^{(i)}$.
- Indistinguishability against the malicious sender. For any PPT adversary S^* and for any $\ell = \text{poly}(\lambda)$, let $\text{View}_{S^*}(S^*, \mathcal{R}(b^{(1)}, b^{(2)}, \dots, b^{(\ell)}))$ denote the view of the adversary when it interacts

⁸We remark that the receiver (resp., sender) has access to fresh randomness for every third (resp., fourth) round message. $r_{\mathcal{R}}$ (resp., r_S) is simply her secret state for the first two messages.

with an honest receiver with inputs $b^{(1)}, b^{(2)}, \ldots, b^{(\ell)}$ (where $b^{(i)}$ denotes the input used by \mathcal{R} in the *i*-th reuse session). It holds that

 $\mathsf{View}_{\mathcal{S}^*}\langle \mathcal{S}^*, \mathcal{R}(b^{(1)}, b^{(2)}, \dots, b^{(\ell)}) \rangle \stackrel{c}{\approx} \mathsf{View}_{\mathcal{S}^*}\langle \mathcal{S}^*, \mathcal{R}(0, 0, \dots, 0) \rangle.$

• Simulation security against the malicious receiver. Same as Definition 3 adapted to the case of oblivious transfer.

5.1 A Building Block

Our construction utilizes a special PKE scheme defined as follows.

Definition 12. We consider a public-key encryption scheme (Gen, Enc, Dec) such that the following hold.

• Correctness. For any message m, it holds that

$$\Pr\begin{bmatrix} (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^{\lambda}), \ \mathsf{ct} = \mathsf{Enc}(\mathsf{pk},m) \\ m' = \mathsf{Dec}(\mathsf{ct},\mathsf{sk}) &: \ m' = m \end{bmatrix} = 1.$$

• Semantic Security. For all PPT adversary A, it holds that

$$\left| \Pr \begin{bmatrix} (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^{\lambda}), \ (m_0,m_1) \leftarrow \mathcal{A}(\mathsf{pk}) \\ b \leftarrow \{0,1\}, \ \mathsf{ct} = \mathsf{Enc}(\mathsf{pk},m_b), \ b' \leftarrow \mathcal{A}(\mathsf{pk},\mathsf{ct}) \\ \end{bmatrix} \cdot \left| b' = b \right| - \frac{1}{2} \right| = \mathsf{negl}(\lambda).$$

Valid public keys are (exponentially) sparse and pseudorandom. Let PK denote the set of all strings in the co-domain of Gen(1^λ). Let PK' be the set of valid public keys, i.e., those public keys in the support of Gen(1^λ). It holds that

 $- |\mathcal{P}\mathcal{K}'|/|\mathcal{P}\mathcal{K}| = 2^{-\mathsf{poly}(\lambda)},$

- For any PPT distinguisher \mathcal{D} , $|\Pr[\mathcal{D}(\mathsf{pk}) = 1] \Pr[\mathcal{D}(U) = 1]| = \mathsf{negl}(\lambda)$, where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^{\lambda})$ and U is sampled uniformly from \mathcal{PK} .
- Invalid public keys statistically hide the message.⁹ For any pk ∉ PK', it holds, for all messages m₀ and m₁, that Enc(pk, m₀) and Enc(pk, m₁) are statistically close.

We provide instantiations of this primitive from a list of assumptions below.

- **DDH.** Such PKE can be constructed from the Decisional Diffie-Hellman assumption (DDH) similar to [NP01]. In particular, $\text{Gen}(1^{\lambda})$ samples a group \mathbb{G} of order p and a generator g. pk and sk are sampled to be $\begin{pmatrix} g & g^a \\ g^b & g^{ab} \end{pmatrix}$ and (a, b) for randomly chosen a and b. The universe \mathcal{PK} is $\begin{pmatrix} g & g^a \\ g^b & g^c \end{pmatrix}$ for all a, b, and c. To encrypt a message $m \in \mathbb{G}$, one sends $g^u \cdot (g^a)^v$ and $m \cdot (g^b)^u \cdot (g^c)^v$. If c = ab, then one can decrypt the message correctly. However, if $c \neq ab$, $g^u \cdot (g^a)^v$ and $m \cdot (g^b)^u \cdot (g^c)^v$ statistically hides m. This construction satisfies all the properties above.
- **SSP-OT.** In general, it can be constructed from any two-round OT protocol (OT_1, OT_2) that satisfies (1) statistical sender privacy (a.k.a. SSP-OT) and (2) that the first round message is pseudorandom. In particular, we could sample the public key to be $(OT_1(0; r_1), \ldots, OT_1(0; r_{\lambda}))$ with independent randomness r_1, \ldots, r_{λ} . To encrypt a message m, one first samples m_1, \ldots, m_{λ} that additively secret shares m. The ciphertext shall be $\{OT_2(ot_{i,1}, (m_i, \bot))\}_{i=1}^{\lambda}$, where $ot_{i,1}$ stands for $OT_1(0; r_i)$.

The valid public key is pseudorandom as the first round message of the OT protocol is pseudorandom. Valid public keys are $2^{-\lambda}$ sparse if we assume the support of the first-round message with choice bit 0 is smaller than that with choice bit 1. Without loss of generality, we may assume this as, otherwise,

 $^{^{9}}$ We note that computational security also works for our construction. We state the statistical security as all of our instantiations enjoys this stronger notion.

we could simply switch the valid public keys to be $(OT_1(1; r_1), \ldots, OT_1(1; r_{\lambda}))$. Finally, invalid public keys statistically hide the message since the OT protocol enjoys statistical sender privacy.

The construction we describe above is a particular case of this general construction using the statistical sender private OT from DDH [NP01].

• **QR.** Recall the PKE scheme [GM82] from the quadratic residuosity problem. Suppose N is a Blum integer $N = p \cdot q$. A random quadratic non-residue x, whose Jacobi symbol $\left(\frac{x}{N}\right)$ is 1, is sampled as the public key. To encrypt a bit b, one samples a random y, which is coprime to N, and encrypt it as $\mathsf{ct} = y^2 \cdot x^b$. The decryption algorithm uses p and q to check if ct is a quadratic residue or not, which, in turn, determines if b is 0 or 1.

This PKE already satisfies several requirements. Valid public key x (i.e., quadratic non-residue with Jacobi symbol 1) is indistinguishable from a random integer with Jacobi symbol 1, hence, is pseudo-random.¹⁰ Invalid public keys statistically hide the message as, if x is a quadratic residue, ct is simply a random quadratic residue, and all information regarding b is lost. The only issue is that the valid public keys are not sparse enough; they are only 1/2-sparse. We use our ideas above again. We sample λ public keys x_1, \ldots, x_{λ} . When one encrypts a bit b, one first additively secret shares b as b_1, \ldots, b_{λ} and encrypts b_i with x_i . This ensures that the valid public keys are $2^{-\lambda}$ -sparse.

5.2 Our Construction

Our protocol is formally presented in Figure 4 using the following building blocks.

- (rEcom₁, rEcom₂, rEcom₃) is a delayed-input reusable extractable commitment scheme from Section 3.5.
- (ZAP_1, ZAP_2) is a ZAP scheme from Section 3.6.
- (Gen, Enc, Dec) is a PKE scheme defined in Section 5.1.
- PRG is a pseudorandom generator with a sufficiently large stretch¹¹ from Section 3.3.
 - Inputs: Suppose there are ℓ reuse sessions (where ℓ is not a priori fixed). Sender holds the input $\{m_0^{(i)}, m_1^{(i)}\}_{i \in [\ell]}$. Receiver holds the input $b^{(1)}, b^{(2)}, \ldots, b^{(\ell)}$.
 - Language \mathcal{L} : Fix (recom_{1,1}, recom_{1,2}) and (recom_{2,1}, recom_{2,2}). A statement

$$\left(\mathsf{recom}_{1,3}^{(i)},\mathsf{recom}_{2,3}^{(i)},r_0^{(i)},r_1^{(i)}\right)$$

is in language \mathcal{L} if there exists a witness $(\alpha^{(i)}, c, \tau)$ such that

- $\left(\mathsf{recom}_{c,1}, \mathsf{recom}_{c,2}, \mathsf{recom}_{c,3}^{(i)}\right)$ is an honest commitment of $\alpha^{(i)}$ with randomness τ ;
- And either $r_0^{(i)}$ or $r_1^{(i)}$ equals to $\mathsf{PRG}(\alpha^{(i)})$.
- Protocol description:
 - 1. Receiver initiates two instances of the delayed-input reusable extractable commitment $\mathsf{recom}_{1,1} \leftarrow \mathsf{rEcom}_1(1^{\lambda};\tau_1)$ and $\mathsf{recom}_{2,1} \leftarrow \mathsf{rEcom}_1(1^{\lambda};\tau_2)$. Receiver sends $\mathsf{recom}_{1,1}, \mathsf{recom}_{2,1}$.

¹⁰Our OT protocol is written assuming pk is pseudorandom over binary strings and, hence, all the operations are over \mathbb{F}_2 . If we plug the QR-based construction into our OT protocol, the operation will be over the multiplicative group \mathbb{J} , i.e., the set of integers with Jacobi 1. Additionally, we need a (deterministic) mapping from a random binary string to a random element from \mathbb{J} (since the PRG outputs are binary strings). For instance, this mapping can be chosen to be any randomized process of picking random elements from \mathbb{J} (i.e., the process uses the input as its randomness to pick elements from \mathbb{J}).

¹¹The stretch we need depends on how sparse the valid public keys are. Looking forward, our proof relies on the fact that strings of the form $\mathsf{pk} \oplus \mathsf{PRG}(s)$ (for all possible valid public keys pk and seed s) are also (exponentially) sparse in the universe. Therefore, if the valid public keys are, for instance, $2^{-\lambda}$ sparse, it suffices to have a PRG of seed length $\leq \lambda/2$ and, consequently, of stretch $\geq \log |\mathcal{PK}|/(\lambda/2)$.

- Sender samples recom_{1,2} ← rEcom₂, recom_{2,2} ← rEcom₂, zap₁ ← ZAP₁, and a random string s. Sender sends (recom_{1,2}, recom_{2,2}, s, zap₁).
- For the i^{th} reuse session,

i-3: Receiver samples random $\alpha^{(i)}$ and $(\mathsf{pk}^{(i)}, \mathsf{sk}^{(i)}) \leftarrow \mathsf{Gen}(1^{\lambda})$. Set

$$r_{b^{(i)}}^{(i)} = s \oplus \mathsf{pk}^{(i)} \qquad \qquad r_{1-b^{(i)}}^{(i)} = \mathsf{PRG}\left(\alpha^{(i)}\right).$$

The receiver commits to $\alpha^{(i)}$ in the first instance of the reusable commitment and junk in the second instance. That is,

$$\begin{aligned} \mathsf{recom}_{1,3}^{(i)} &= \mathsf{rEcom}_3(\tau_1,\mathsf{recom}_{1,2},\alpha^{(i)}), \\ \mathsf{recom}_{2,3}^{(i)} &= \mathsf{rEcom}_3(\tau_2,\mathsf{recom}_{2,2},\bot). \end{aligned}$$

The receiver samples $zap_2^{(i)}$ which uses the witness $(\alpha^{(i)}, 1, \tau_1)$ to prove

$$(\mathsf{recom}_{1,3}^{(i)},\mathsf{recom}_{2,3}^{(i)},r_0^{(i)},r_1^{(i)})\in\mathcal{L}.$$

 $\underline{\text{Receiver sends}}\left(\mathsf{recom}_{1,3}^{(i)},\mathsf{recom}_{2,3}^{(i)},r_{1}^{(i)},\mathsf{zap}_{2}^{(i)}\right) \text{ to the sender.}$

- *i*-4: Sender verifies that $(\text{recom}_{1,1}, \text{recom}_{1,2}, \text{recom}_{1,3})$, $(\text{recom}_{2,1}, \text{recom}_{2,2}, \text{recom}_{2,3})$, and $\text{zap}_1, \text{zap}_2^{(i)}$ are all accepting. If the verification fails, the sender aborts. For both $j \in \{0, 1\}$, set $\mathsf{pk}_j^{(i)} = s \oplus r_j^{(i)}$ and $\mathsf{ct}_j^{(i)} = \mathsf{Enc}(\mathsf{pk}_j^{(i)}, m_j^{(i)})$. Sender sends $\underbrace{(\mathsf{ct}_0^{(i)}, \mathsf{ct}_1^{(i)})}$ to the receiver.
- Receiver Output. The receiver recovers message $m_{b^{(i)}}^{(i)} = \text{Dec}(\mathsf{sk}^{(i)}, \mathsf{ct}_{b^{(i)}}^{(i)})$.

Figure 4: Our four-round reusable OT

We shall prove the following theorem.

Theorem 4. Assuming the security of the reusable extractable commitment scheme, the ZAP scheme, the pseudorandom generator, and the PKE scheme, the protocol defined in Figure 4 is a reusable four-round oblivious transfer protocol with simulation security against malicious receivers and indistinguishability security against malicious senders (see Definition 11).

The correctness of protocol follows the correctness of the PKE scheme.

5.3 Indistinguishability against a malicious sender.

We prove it through a sequence of hybrids. Note that H_0 is the sender's view when he interacts with an honest receiver. And the last hybrid is the sender's view when the honest receiver's inputs are fixed to be $(0, 0, \ldots, 0)$.

Hybrid Description.

- H_0 : Sender's view in the real world.
- H_1 : This hybrid is identical to the previous one except that $r_{b^{(i)}}^{(i)}$ is sampled uniformly at random.
- H_2 : This hybrid is identical to the previous one except that $r_{b^{(i)}}^{(i)}$ is sampled as $\mathsf{PRG}(\hat{\alpha}^{(i)})$ for a random $\hat{\alpha}^{(i)}$.
- H_3 : This hybrid is identical to the previous one except that $\operatorname{recom}_{2,3}$ now also commits to $\alpha^{(i)}$. That is, $\operatorname{recom}_{2,3}^{(i)} = \operatorname{rEcom}_3(\tau_2, \operatorname{recom}_{2,2}, \alpha^{(i)})$.

- H_4 : This hybrid is identical to the previous one except that ZAP uses the second instance of the commitment as the witness. That is, $zap_2^{(i)}$ uses the witness $(\alpha^{(i)}, 2, \tau_2)$ to prove $(\mathsf{recom}_{1,3}^{(i)}, \mathsf{recom}_{2,3}^{(i)}, r_1^{(i)}) \in \mathcal{L}$.
- H₅: This hybrid is identical to the previous one except that recom_{1,3} now commits to either α⁽ⁱ⁾ or â⁽ⁱ⁾ depending on r₁⁽ⁱ⁾. That is, if r₁⁽ⁱ⁾ = PRG(α⁽ⁱ⁾), then recom_{1,3}⁽ⁱ⁾ = rEcom₃(τ₁, recom_{1,2}, α⁽ⁱ⁾); otherwise, recom_{1,3}⁽ⁱ⁾ = rEcom₃(τ₁, recom_{1,2}, â⁽ⁱ⁾).
- H_6 : This hybrid is identical to the previous one except that ZAP uses the first instance of the commitment as the witness. That is, $zap_2^{(i)}$ uses the witness either $(\alpha^{(i)}, 1, \tau_1)$ or $(\widehat{\alpha}^{(i)}, 1, \tau_1)^{12}$ to prove $(recom_{1,3}^{(i)}, recom_{2,3}^{(i)}, r_1^{(i)}) \in \mathcal{L}$.
- H_7 : This hybrid is identical to the previous one except that $\mathsf{recom}_{2,3}$ now switches back to a commitment of \bot .
- H_8 : This hybrid is identical to the previous one except that $r_0^{(i)}$ is sampled uniformly at random.
- H_9 : This hybrid is identical to the previous one except that $r_0^{(i)}$ is sampled as $s \oplus \mathsf{pk}^{(i)}$ where $\mathsf{pk}^{(i)}$ is sampled as the output of $\mathsf{Gen}(1^{\lambda})$.

Security Proof. We prove the indistinguishability of hybrids below.

Claim 7. Assuming the valid public keys are pseudorandom, H_0 and H_1 are indistinguishable.

Proof. Assume for the sake of contradiction that H_0 and H_1 are computationally distinguishable with a non-negligible advantage. We show that this contradicts the pseudorandomness of the public keys property of our PKE scheme.

The simulator simply simulates the hybrid as described until it sends the third-round message. When it needs to send $r_{b^{(i)}}^{(i)}$, it receives a string x from the external challenger, which is either a valid public key or a random string. It sends $s \oplus x$ as $r_{b^{(i)}}^{(i)}$. Depending on whether x is a valid public key or a random string, it either simulates H_0 or H_1 . If there is a distinguisher that can distinguish between H_0 and H_1 with a non-negligible advantage, then it contradicts the pseudorandomness of public keys.

Claim 8. Assuming the pseudorandomness of the PRG, H_1 and H_2 are indistinguishable.

Proof. This proof is similar to the previous one. The simulator simply simulates the hybrid as described. When it needs to send $r_{b^{(i)}}^{(i)}$, it receives a x string from the external challenger, which is either the output of the PRG on uniform input or a random string. It sends x as $r_{b^{(i)}}^{(i)}$. Clearly, it either simulates H_1 or H_2 . Hence, H_1 and H_2 are indistinguishable by the pseudorandomness of the PRG.

Claim 9. Assuming the hiding property of the reusable delayed-input extractable commitment schemes, H_2 and H_3 are indistinguishable.

Proof. The simulator samples $\alpha^{(i)}$. It sends to the external challenger $(\alpha^{(1)}, \ldots, \alpha^{(\ell)})$ and (\perp, \ldots, \perp) . The simulator and the external challenger engage in a reusable delayed-input extractable commitment scheme, where the external challenger commits to either $(\alpha^{(1)}, \ldots, \alpha^{(\ell)})$ or (\perp, \ldots, \perp) . The simulator now simulates the hybrid as described. It sends the external challenger's messages in place of the second instance of the commitment. Since the other components of the protocol do not depend on the private randomness of the second commitment scheme, the simulator can simulate the hybrid. Depending on whether the external challenger commits to $(\alpha^{(1)}, \ldots, \alpha^{(\ell)})$ or (\perp, \ldots, \perp) , it either simulates H_2 or H_3 . By the hiding property, H_2 and H_3 are indistinguishable.

Claim 10. Assuming the witness indistinguishability of the ZAP scheme, H_3 and H_4 are indistinguishable.

¹²depending on whether $\mathsf{recom}_{1,3}^{(i)}$ is a commitment of $\alpha^{(i)}$ or $\widehat{\alpha}^{(i)}$.

Proof. If the two hybrids are distinguishable with a non-negligible advantage, then we show a simulator that breaks the witness indistinguishability of the ZAP scheme. The simulator simulates the hybrid as described. It sends the zap_1 in the hybrid to the external challenger as the first message of the ZAP scheme. It also sends the statement and two witnesses to the external challenger. It forwards the proof it receives from the external challenger as $zap_2^{(i)}$ in the hybrid. Depending on which witness the external challenger uses, it simulates either H_3 or H_4 . Therefore, the witness indistinguishability of the ZAP scheme implies that H_3 and H_4 are indistinguishable.

Claim 11. Assuming the hiding property of the reusable delayed-input extractable commitment schemes, H_4 and H_5 are indistinguishable.

Proof. This proof is analogous to the proof of Claim 9. Observe that all the ZAP proofs are on the second instance of the commitment scheme and, hence, the first instance is independent of the other components in the protocol. \Box

Claim 12. Assuming the witness indistinguishability of the ZAP scheme, H_5 and H_6 are indistinguishable.

Proof. This proof is analogous to the proof of Claim 10.

Claim 13. Assuming the hiding property of the reusable delayed-input extractable commitment schemes, H_6 and H_7 are indistinguishable.

Proof. This proof is analogous to the proof of Claim 9. Observe that all the ZAP proofs are back on the first instance of the commitment scheme and, hence, the second instance is independent of the other components in the protocol. \Box

Claim 14. Assuming the pseudorandomness of the PRG, H_7 and H_8 are indistinguishable.

Proof. The proof is analogous to the proof of Claim 8.

Claim 15. Assuming the valid public keys are pseudorandom, H_8 and H_9 are indistinguishable.

Proof. The proof is analogous to the proof of Claim 7.

Remark 3 (Public recovery of the received message in the reusable setting). Later in our MPC protocol, every pair of parties shall engage in an instance of a reusable OT protocol. Furthermore, in the last round, the receiver of the OT protocol needs to publish her private randomness to help all parties recover the message she receives. This has to be done without harming the reusability of the OT protocol.

Observe that, in our OT protocol, the receiver could publish the randomness used in

$$(\mathsf{pk}^{(i)},\mathsf{sk}^{(i)}) \leftarrow \mathsf{Gen}(1^{\lambda}).$$

This helps all parties to decrypt the message it received in the *i*th reuse session. Moreover, it is easy to see that the receiver's choice bits for other reuse sessions remain hidden. We remark that the indistinguishability of the receiver's choice bit in the *i*th session holds even if the sender learns $pk^{(j)}, sk^{(j)}$ for all $j \neq i$. We make a note of this as when we use the reusable OT as a building block in our MPC protocol, we do need the receiver to send out $pk^{(j)}, sk^{(j)}$ in the fourth round (in the simultaneous message exchange model) to help other parties recover the message the receivers.

5.4 Simulation security for a malicious receiver.

The simulator is presented in Figure 5.

- Run the first three rounds and check abort. In the main thread, it runs the protocol honestly for all reuse sessions until the first time the receiver's third round message does not cause an abort. (Observe that at this point, all sender's messages are independent of its inputs.)
- Rewind to extract k_1 and k_2 . The simulator rewinds back to the end of the first round and generates a set of lookahead threads (running until round 3). It uses the EcomExt to extract the key k_1 and k_2 committed in the first rounds of the two instances of the reusable delayed-

input extractable commitment scheme. Recall that in our construction of reusable delayed-input extractable commitment, the extractable commitment is used to commit to a key for an SKE scheme. If the extraction fails, it aborts.

- Extract choice bits and query the ideal functionality. For every session such that the third round is accepting, it uses the key k_1 and k_2 to extract the committed messages $\alpha^{(i)}$ and $\hat{\alpha}^{(i)}$. If $r_0^{(i)}$ equals to either $\mathsf{PRG}(\alpha^{(i)})$ or $\mathsf{PRG}(\hat{\alpha}^{(i)})$, it set $b^{(i)}$ to be 1. Else if $r_1^{(i)}$ equals to either $\mathsf{PRG}(\alpha^{(i)})$, it set $b^{(i)}$ to be 0. Otherwise, it aborts. If $b^{(i)}$ is found, send it to the ideal functionality and receives $m_{b^{(i)}}^{(i)}$.
- Finish the fourth round by encrypting the same message. The simulator now finishes the main thread by computing and sending

$$\left(\mathsf{Enc}(\mathsf{pk}_0^{(i)}, m_{b^{(i)}}^{(i)}), \ \mathsf{Enc}(\mathsf{pk}_1^{(i)}, m_{b^{(i)}}^{(i)})\right)$$

as the fourth round message.

Figure 5: The simulator for a malicious receiver

We shall prove it via a sequence of hybrids. The first hybrid H_0 is the real world and the last hybrid is the ideal world.

Hybrid Description.

- H_0 : This is the real world.
- (Extracting keys k_0, k_1) H_1 : If the verification for the third round fails in all sessions, the simulator aborts as in the previous hybrid. Otherwise, the simulator rewinds back to the end of the first round. It generates a number of lookahead threads that run the second and third rounds to extract the committed keys k_1 and k_2 .¹³ We do not make any other changes to the main thread and, hence, this hybrid is identically distributed to the previous one.
- (Queries the ideal functionality) H_2 : In this hybrid, for every accepting third-round message, the simulator uses k_0 and k_1 to extract the receiver's choice bits $b^{(i)}$. In particular, for the i^{th} session, it does the following. If the verification for third round succeeds, it first extracts the committed messages $\alpha^{(i)}$ and $\hat{\alpha}^{(i)}$. If $r_0^{(i)}$ equals to either $\mathsf{PRG}(\alpha^{(i)})$ or $\mathsf{PRG}(\hat{\alpha}^{(i)})$, it set $b^{(i)}$ to be 1. Else if $r_1^{(i)}$ equals to either $\mathsf{PRG}(\alpha^{(i)})$, it set $b^{(i)}$ to be 0. Otherwise, it aborts. The simulator sends the choice bits $b^{(i)}$ to the ideal functionality and receives $m_{b^{(i)}}^{(i)}$.
- (Encrypt the same message in both ciphertexts) $H_{3,j}$: This hybrid is identical to the previous hybrid in all the i > j sessions. For the other sessions (i.e., $i \leq j$), in the *i*-4 round, it sends $\operatorname{Enc}(\mathsf{pk}_0^{(i)}, m_{b^{(i)}}^{(i)})$ and $\operatorname{Enc}(\mathsf{pk}_1^{(i)}, m_{b^{(i)}}^{(i)})$ (instead of sending $\operatorname{Enc}(\mathsf{pk}_0^{(i)}, m_0^{(i)})$ and $\operatorname{Enc}(\mathsf{pk}_1^{(i)}, m_1^{(i)})$).

Security Proof.

Claim 16. Assuming the soundness of ZAP, H_2 is indistinguishable from H_1 .

Proof. The only difference between H_2 and H_1 is when there exists an *i* such that $b^{(i)}$ does not exist. Assuming H_2 and H_1 are not indistinguishable, it implies that there exists an *i* such that the third round message is accepting, but $b^{(i)}$ does not exist with non-negligible probability. We shall break the soundness of ZAP as follows.

The simulator engages in a ZAP protocol with the honest verifier. It receives the first round message r from the verifier and proceeds to simulate the hybrid as described. It sends the r it received in place of

¹³Recall in the reusable delayed-input extractable commitment scheme, a key for the PRF is committed in the first round. Here, k_1 (resp., k_2) is the key committed in the first (resp., second) instance of the commitment scheme.

 zap_1 . Now, it receives back a statement $st = \left(recom_{1,3}^{(i)}, recom_{2,3}^{(i)}, r_1^{(i)}\right)$ and a proof $\pi = zap_2^{(i)}$. Since our extraction of $b^{(i)}$ fails, we now argue that it must be the case $st \notin \mathcal{L}$ with overwhelming probability. Specifically, if both the commitments are invalid then the statement $st \notin \mathcal{L}$. Otherwise, if at least one of the commitments is valid, then the extractor for the extractable commitment extracts this committed value except with negligible probability. In this case, if both $r_0^{(i)}$ and $r_1^{(i)}$ is not equal to the PRG applied on the extracted value, then $st \notin \mathcal{L}$. Therefore, with non-negligible probability, the simulator generates a false statement st but an accepting proof π . Thus, it breaks the soundness of ZAP.

Claim 17. $H_{3,j}$ is (statistically) indistinguishable from $H_{3,j-1}$.

Proof. This claim follows from the following claim.

Claim 18. With high probability over the choice of s, for all α , $s \oplus \mathsf{PRG}(\alpha)$ is not a valid public key.

If this claim is true, then $H_{3,j}$ is clearly (statistically) indistinguishable from $H_{3,j-1}$ as invalid public keys statistically hide the message. That is,

$$\mathsf{Enc}(\mathsf{pk}_{1-b^{(j)}}^{(j)}, m_{1-b^{(j)}}^{(j)}) \approx \mathsf{Enc}(\mathsf{pk}_{1-b^{(j)}}^{(j)}, m_{b^{(j)}}^{(j)}),$$

since $\mathsf{pk}_{1-b^{(j)}}^{(j)} = s \oplus \mathsf{PRG}(\alpha)$ for some α .

The proof of this claim is similar to the proof of the (statistical) binding property of Naor's commitment scheme [Nao91]. Call an *s* "bad" if there exists a valid public key pk and an α such that $s = \mathsf{pk} \oplus \mathsf{PRG}(\alpha)$. The number of "bad" *s* is upper bounded by the product of (1) the number of valid public keys and (2) the input size of the PRG. Recall that our PKE scheme is required to have scarce valid public keys. Hence, for a PRG with a sufficiently large stretch, the number of "bad" *s* shall be only a negligible fraction of the size of the universe. Therefore, most choices of *s* are "good" and the claim is proven.

6 Reusable 2PC Protocol

In this section, we shall use our reusable OT protocol and the reusable zero-knowledge protocol to construct a reusable 2PC protocol. We consider the unidirectional message setting. That is, for every round, only one party shall send messages to the other party. In this setting, we consider a four-round reusable 2PC protocol, where only one party receives the output of the protocol.¹⁴ In our protocol, the first two rounds shall only be executed once. In the third round, the receiver shall send a message depending on his input x. In the last round, the sender sends one message depending on her function f. Parties could either reuse the first two rounds, where both the input x and the function f could change, or reuse the first three rounds, where only the function f could change.

Our protocol follows the standard 2PC protocol based on Yao's garbled circuit. That is, a four-round oblivious transfer protocol is run and in the last round, the sender shall generate a garbled circuit and send the receiver his labels using the OT protocol. Since our OT protocol supports reusing its first two rounds with different receiver's choice bits, our 2PC protocol also supports reusing the first two rounds with different receiver's inputs. Additionally, we parallel this with our ZK-AoK protocol where the sender proves that the garbled circuits and OT messages are generated honestly. Crucially, we also rely on that the zero-knowledge protocol is reusable so that the sender could prove a different statement in every fourth round message.

Our protocol is formally presented in Figure 6 with the following blocks.

- $(rOT_1, rOT_2, rOT_3, rOT_4)$ is a four-round reusable OT from Section 5.
- $(rZK_1, rZK_2, rZK_3, rZK_4)$ is a four-round reusable ZK-AoK from Section 4.
- (Garble, Eval) is a garbling scheme from Section 3.4.

We use C for the universal circuit that takes the input a function f and an input x and outputs f(x).

 $^{^{14}}$ This is optimal as Katz and Ostrovsky [KO04] proved that five rounds are needed if both parties shall receive the output of the protocol.

Remark 4. As we discussed in the technical overview, it is necessary that the sender maintains (and updates) a secret state across different reuse sessions. Otherwise, a malicious receiver could employ the same strategy as the (black-box) simulator to effectively "rewind" the sender across multiple reuse sessions and, hence, extract the input of the sender.

• Language $\mathcal{L}^{(i)}$. Fix the first three rounds $(\mathsf{rot}_1, \mathsf{rot}_2, \mathsf{rot}_3^{(i)})$ and $(\mathsf{rzk}_1, \mathsf{rzk}_2, \mathsf{rzk}_3)$. A statement

$$\mathsf{st}^{(i)} = \left(\widehat{C}, \; \{\mathsf{lab}_{w, f_w^{(i)}}\}_{w \in \mathcal{S}}, \; \mathsf{rot}_4^{(i)}\right)$$

is in the language $\mathcal{L}^{(i)}$ with witness $(f^{(i)}, r^{(i)}, r_{\mathcal{S}, \mathsf{rot}}, r_{\mathcal{S}, \mathsf{rot}}^{(i)})$ if both the following conditions are satisfied.

- \hat{C} an honest garbling of $\text{Garble}(C; r^{(i)})$ and $\{\text{lab}_{w, f_w^{(i)}}\}_{w \in S}$ is the correct labels corresponding to the sender's input $f^{(i)}$.
- $\operatorname{rot}_{4}^{(i)}$ is honestly generated with the partial transcript $(\operatorname{rot}_1, \operatorname{rot}_2, \operatorname{rot}_3^{(i)})$ and the receiver's input labels as the input messages using randomness $(r_{S, \operatorname{rot}}, r_{S, \operatorname{rot}}^{(i)})$.
- Protocol Description.
 - 1. The receiver sends $\mathsf{rot}_1 \leftarrow \mathsf{rOT}_1(1^\lambda; r_{\mathcal{R},\mathsf{rot}})$ and $\mathsf{rzk}_1 \leftarrow \mathsf{rZK}_1(1^\lambda; r_{\mathcal{R},\mathsf{rzk}})$.
 - 2. The sender sends $\mathsf{rot}_2 \leftarrow \mathsf{rOT}_2(1^{\lambda}; r_{\mathcal{S},\mathsf{rot}} \text{ and } \mathsf{rzk}_2 \leftarrow \mathsf{rZK}_2(1^{\lambda}; r_{\mathcal{S},\mathsf{rzk}}).$
 - Suppose the sender and receiver are going to execute a new reuse session with $f^{(i)}$ and $x^{(i)}$ as their inputs, respectively.
 - *i*-3. The receiver samples $\operatorname{rot}_3^{(i)} \leftarrow \operatorname{rOT}_3(\operatorname{rot}_2, x^{(i)}, r_{\mathcal{R}, \operatorname{rot}})$. That is, the receiver computes the third-round message of the reusable OT with $x^{(i)}$ as his choice bits. The receiver computes $\operatorname{rzk}_3 \leftarrow \operatorname{rZK}_3(\operatorname{rzk}_2, r_{\mathcal{R}, \operatorname{rzk}})$. It sends $\operatorname{rot}_3^{(i)}$ and rzk_3 .
 - *i*-4. Secret state updates. The sender maintains a secret state ω , which is initially set to be an empty string. If ω is empty, it updates ω to be rzk_3 , i.e., this is the first time the sender receives the third round message rzk_3 . If ω is not empty and $\omega \neq rzk_3$, the sender aborts, i.e., the new rzk_3 message the sender receives is different from the ones it receives from previous reuse sessions.

The sender samples $(\widehat{C}, \{\mathsf{lab}_{w,b}\}_{w,b}) \leftarrow \mathsf{Garble}(C; r^{(i)})$ and $\mathsf{rot}_4^{(i)} = \mathsf{rOT}_4(\mathsf{rot}_1, \mathsf{rot}_3^{(i)}, \{\mathsf{lab}_{w,0}\}_{w\in\mathcal{R}}, \{\mathsf{lab}_{w,1}\}_{w\in\mathcal{R}}, r_{\mathcal{S},\mathsf{rot}}; r_{\mathcal{S},\mathsf{rot}}^{(i)})$. That is, she samples the fourth OT message with the labels for the receiver's input wire as her messages. She samples $\mathsf{rzk}_4^{(i)} \leftarrow \mathsf{rZK}_4(\mathsf{rzk}_3, r_{\mathcal{S}, r_{\mathsf{rzk}}}, \mathsf{st}^{(i)}, (f^{(i)}, r^{(i)}), r_{\mathcal{S},\mathsf{rot}}, r_{\mathcal{S},\mathsf{rot}}^{(i)})$, where $\mathsf{st}^{(i)} =$

 $(\widehat{C}, \{\mathsf{lab}_{w, f_w^{(i)}}\}_{w \in \mathcal{S}}, \mathsf{rot}_4^{(i)}) \text{ and } (f^{(i)}, r^{(i)}, r_{\mathcal{S}, \mathsf{rot}}, r_{\mathcal{S}, \mathsf{rot}}^{(i)}) \text{ is the witness. } \underline{\text{The sender sends } \widehat{C}, } \\ \underline{\{\mathsf{lab}_{w, f_w^{(i)}}\}_{w \in \mathcal{S}}, \mathsf{rot}_4^{(i)}, \text{ and } \mathsf{rzk}_4^{(i)}.}$

- Suppose the sender and receiver are going to use an old reuse session. In this case, only the sender needs to compute and send a new fourth-round message (w.r.t. a new input function f'). The sender does exactly the same as in *i*-4 except that she does not need to do the "secret state update" step.
- The receiver output. It verifies that (rzk₁, rzk₂, rzk₃, rzk₄⁽ⁱ⁾) is an accepting proof of the statement st⁽ⁱ⁾. If not, it aborts. Otherwise, it recovers {lab_{w,x_w}}_{w∈R} from the reusable OT. Then, it evaluates z⁽ⁱ⁾ = Eval(Ĉ, {lab_{w,b}}_{w,b}). It outputs z⁽ⁱ⁾ as C(f⁽ⁱ⁾, x⁽ⁱ⁾).

Figure 6: Our reusable 2PC protocol in the unidirectional message model

We shall prove the following.

Theorem 5. Assuming the security of the reusable OT, the reusable ZK-AoK, and the garbling scheme, the protocol in Figure 6 is a simulation-secure reusable 2PC protocol.

6.1 The Sender is Corrupt

The simulator for the sender is presented in Figure 7. We shall prove indistinguishability via a sequence of indistinguishable hybrids.

- The simulator interacts with the malicious sender in a full execution as an honest receiver with input **0** in each reuse session. It continues the execution until it finds the first reuse session where the fourth round message sent by the sender has an accepting proof (for the non-accepting sessions, it forwards \perp as the output of the honest receiver). The simulator fixes the first two rounds of this protocol and invokes the proof of knowledge extractor for this session. From Figure 3, we infer that this knowledge extractor rewinds and extracts the key k committed in extractable commitment which is part of the reusable delayed-input extractable commitment scheme. The simulator outputs the view of the adversary in the first two rounds.
- For each reuse session, the simulator again uses $\mathbf{0}$ as the honest receiver input and continues with the execution. If the fourth round message sent by the sender is not accepting, it instructs the ideal functionality to abort. Otherwise, it uses the key k to extract the sender's input in this rewind session. It then sends the sender's input to the ideal functionality and instructs it to send the output to the receiver.

Figure 7: Simulator for the sender

Hybrid Description.

- H_0 : This is the real world.
- H_1 : This hybrid is identical to the previous one except for the following. If the sender sends a valid fourth-round message, we invoke the knowledge extractor to extract the sender's input as described in the simulation and use this to compute the output of the receiver.
- H_2 : This hybrid is identical to the previous one except that we now switch the receiver's input from $x^{(i)}$ to **0**.

Security Proof.

Claim 19. Assuming the knowledge extraction property of the reusable ZK-AoK, H_0 and H_1 are indistinguishable.

Proof. The only difference between H_0 and H_1 is when the knowledge extraction fails. Let us assume without loss of generality that the adversary outputs a valid fourth-round message with non-negligible probability. Otherwise, the proof sent in the fourth round message is not accepting with overwhelming probability and, hence, the output of the honest receiver is \perp . Thus, if the extraction fails with non-negligible probability, we shall break the knowledge extractor property of the reusable ZK-AoK. In particular, the prover \mathcal{A} shall interact with a knowledge extractor. It simulates the hybrid exactly as described except for the ZK-AoK protocol, which is exposed to the external knowledge extractor. If the two hybrids are not indistinguishable, it implies that the prover sends an accepting proof with non-negligible probability, but the extractor fails to extract a valid witness with non-negligible probability, then this breaks the knowledge extraction and this is a contradiction.

Claim 20. Assuming the receiver indistinguishability of the reusable OT, H_1 and H_2 are indistinguishable.

Proof. Suppose H_1 and H_2 are distinguishable with a non-negligible advantage, then we show how to break the receiver indistinguishability of the reusable OT by designing a corrupt sender S^* .

The sender S^* generates the verifier messages of the ZK-AoK on its own. It receives the rOT messages from the challenge receiver \mathcal{R} . It sends the ZK-AoK messages and the rOT messages to the malicious sender. It then forward the rOT messages it receives from the sender to the challenge receiver. Depending on whether the receiver's input is $x^{(i)}$ or $\mathbf{0}$, it either simulates H_1 or H_2 . Hence, it breaks the receiver indistinguishability of the reusable OT if H_1 and H_2 are distinguishable.

6.2 The Receiver is Corrupt

The simulator for the receiver is presented in Figure 8. We shall prove indistinguishability via a sequence of hybrids.

- The simulator interacts with the malicious receiver by invoking the simulator for the rOT and the rZK protocols until it finds a reuse session where the receiver sends a valid third round message. It fixes the first round message of the protocol and as described in Figure 2, it rewinds the second and third rounds and extracts the trapdoor t^* . Additionally, it extracts the key k_1 and k_2 committed in the extractable commitments used as part of the rOT protocol (as described in Figure 5). It outputs the view of the adversary in the first two rounds of the protocol.
- For every reuse session:
 - Extract the receiver's input. If the receiver's third-round message is not valid, it aborts. Otherwise, it shall invoke the extractor for the OT protocol (as described in Figure 5) that uses the keys k_1 and k_2 to extract the receiver's input $x^{(i)}$.
 - Query the ideal functionality. The simulator queries the ideal functionality with $x^{(i)}$ and receives $z^{(i)}$ back.
 - Simulate the ZK proof. The simulator generates the ZK proof by invoking the zeroknowledge simulator that uses the trapdoor t^* .
 - Simulate the Garbled Circuits. The simulator generates the garble circuits using the simulator, i.e., $(\widehat{C}, \{\mathsf{lab}_w\}_{w \in \mathcal{R} \cup \mathcal{S}}) \leftarrow \mathsf{Sim}(1^{\lambda}, 1^{|f^{(i)}|}, z^{(i)}).$
 - Finally, the simulator prepares the $rot_4^{(i)}$ using $\{lab_w\}_{w\in\mathcal{R}}$ as the output from the OT functionality.

Figure 8: Simulator for The Receiver

Hybrid Description.

- H_0 : This is the real world.
- H_1 : This hybrid is identical to the previous ones except that the simulator invokes the simulator for zero-knowledge to generate $\mathsf{rzk}_4^{(i)}$.
- H_2 : This hybrid is identical to the previous one except that if the receiver's third round message is valid, we shall rewind to extract the receiver's input using the OT extractor for the receiver's choice bits. It also invokes the OT simulator for the receiver to prepare $rot_4^{(i)}$.
- H_3 : This hybrid is identical to the previous ones except that the simulator invokes the simulator for the garble circuits. Specifically, we query the ideal functionality on input $x^{(i)}$ and obtain $z^{(i)}$ and we invoke the garbled circuits simulator on $z^{(i)}$ to generate $(\widehat{C}, \{\mathsf{lab}_w\}_{w \in \mathcal{R} \cup \mathcal{S}})$. We use this to generate the fourth round message in the protocol.

We note that the indistinguishability of H_2 and H_3 follows directly from the simulation security of garbled circuits. Further, we observe that H_3 is distributed identically to the ideal execution using Sim described in Figure 8.

Security Proof.

Claim 21. Assuming the zero-knowledge property of the reusable ZK-AoK, H_0 and H_1 are indistinguishable.

Proof. If H_0 and H_1 are computationally distinguishable, we could use the malicious receiver to construct a malicious verifier and use D to construct a distinguisher that breaks the zero-knowledge property of the reusable ZK-AoK. Specifically, the malicious receiver simulates the hybrid as described except for the ZK-AoK messages, which are exposed to the external prover. In each reuse session, the statement that is given by the distinguisher is $\mathfrak{st}^{(i)}$ and the witness corresponds to $(f^{(i)}, r^{(i)}, r_{\mathcal{S}, \mathsf{rot}}, r_{\mathcal{S}, \mathsf{rot}}^{(i)})$. If the prover messages generated by the external challenger are generated as the honest prover messages, then we perfectly emulate H_0 and otherwise, the challenger generates the message using the simulator for the ZK-AoK, and we emulate H_1 . Thus, if H_0 and H_1 are distinguishable, it contradicts the zero-knowledge property of reusable ZK-AoK. Note that this claim requires that the rzk_3 message is fixed across all reuse sessions as the zero-knowledge property only holds if the third round message is fixed.

Claim 22. Assuming simulation security against corrupt receivers of the reusable OT, H_1 is indistinguishable from H_2 .

Proof. If H_1 and H_2 are not indistinguishable, we could use the malicious receiver to construct a receiver and a distinguisher that breaks the simulation security against corrupt receivers of the reusable OT. Specifically, in each reuse session, we use the malicious receiver to generate the receiver OT message. The distinguisher constructs the garbled circuit \tilde{C} and provides $\{lab_{w,0}, lab_{w,1}\}_{w\in S}$ as the honest sender inputs for this reuse session. We use the sender rOT messages generated by the challenger and compute the rest of the protocol messages as in the previous hybrid and forward this to the corrupt receiver. The reduction finally runs the distinguisher D on the view of the malicious receiver and outputs whatever D outputs.

If the messages generated by the challenger correspond to the real execution, the output of the above reduction is identical to H_1 . Else, it is identical to H_2 . This contradicts the simulation security against corrupt receivers.

7 Reusable MPC Protocol

In this section, we shall use our reusable oblivious transfer protocol to construct a four-round reusable MPC protocol. That is, the first two rounds of the protocol are executed once. Afterward, given every new input \vec{x} , parties execute the third round in a new reuse session. Finally, given any function f for a reuse session, parties execute the fourth round to evaluate $f(\vec{x})$.

Recently, Choudhuri et al. $[CCG^+20]$ constructed a four-round MPC protocol using any four-round OT protocol. Our protocol adapts their protocol appropriately to make it reusable. In particular, we instantiate the building blocks in their protocol using appropriate reusable variants. For instance, we instantiate the four-round OT in their protocol with the reusable four-round OT that we construct in Section 5.

Intuitively, Choudhuri et al. [CCG⁺20] prove the security of their protocol through a sequence of hybrids, where the indistinguishability among the hybrids reduces to the security of the underlying building blocks. We observe that, as long as each building block supports reusable security, their proof also works to prove the reusable security of the protocol by the same reduction.

We present the protocol in Section 7.2. The simulator and a formal security proof are included in Section 7.3 and Section 7.4.

7.1 Additional Building Blocks

7.1.1 A Rewind-secure Extractable Commitment Scheme for Reusable MPC

As in [CCG⁺20], we use the following variant of the reusable extractable commitment scheme. Note that this commitment scheme is *reusable rewind-secure*. That is, the committed message is hidden even if the receiver could rewind the sender and ask for multiple third-round messages. For technical reasons, we shall prove and use the rewind security in the proof directly. For our purpose, we shall set B = 4 in our construction. Note that if B = 4, one could extract the committed polynomial using any 5 well-formed transcripts.

A prominent feature of this commitment scheme is that the receiver cannot verify the well-formedness of the sender's third-round message. Therefore, one must rely on additional proof to be convinced if the transcript is well-formed or not.

- Commitment Phase. Let Com be a non-interactive commitment scheme.
 - 1. Sender picks N degree-B random polynomials p_1, p_2, \ldots, p_N over \mathbb{F}_q , where $q > 2^{\lambda}$ is a prime. For all $j \in [N]$, Sender computes $\mathsf{recom}_{1,j} = \mathsf{Com}(p_j; r_j)$. Sender sends $\mathsf{recom}_1 = (\mathsf{recom}_{1,1}, \ldots, \mathsf{recom}_{1,N})$

The sender also sample a random key k. This key k is not used in the first round, but it is fixed for all reuse sessions.

- 2. For all $j \in [N]$, receiver samples a random z_j from \mathbb{F}_q^* . Receiver sends $\mathsf{recom}_2 = (z_1, \ldots, z_N)$.
- 3. **Reuse.** Suppose the sender wants to commit to $m^{(i)}$ in the i^{th} reuse session.
 - For all $j \in [N]$, computes $\mathsf{recom}_{3,j} = (k \oplus p_j(0), p_j(z_j))$.
 - Let $\operatorname{recom}_{3,N+1} = \mathsf{SKE}.\mathsf{Enc}(k, m^{(i)}).$
 - Sends $\operatorname{recom}_3 = (\operatorname{recom}_{3,1}, \dots, \operatorname{recom}_{3,N}, \operatorname{recom}_{3,N+1}).$
- **Decommitment.** To decommit, the sender simply sends all polynomials p_1, \ldots, p_N together with the private randomness r_1, \ldots, r_N .

Figure 9: (Reusable) Rewind-secure Extractable Commitment

7.1.2 (Reusable) Non-malleable Commitment.

We import the following definitions and theorems from $[CCG^+20]$ regarding non-malleable commitment.

Consider a three-round commitment scheme. $[CCG^+20]$ consider the following notion of *non-malleability* with respect to extraction. In the experiment, a Man-In-the-Middle (MIM) adversary interacts with an honest sender in the left session and interacts with an extractor Ext_{NMCom} on the right session, which only guarantees the extraction of the committed value when the adversary outputs a well-formed commitment. Without loss of generality, we assume the left session and right session are associated with distinct tags/identities. The honest sender on the left holds an input m and the MIM attacker holds an auxiliary input z. Let $MIM_{(C,Ext_{NMCom})}^{Ext}(m, z)$ stands for the joint distribution of the extracted value val and the adversary's view.

Definition 13. A three-round commitment scheme is said to be non-malleable with respect to extraction if there exists an extractor Ext_{NMCom} such that

$$\mathsf{MIM}_{\langle C,\mathsf{Ext}_{\mathsf{NMCom}}\rangle}^{\mathsf{Ext}}(m_0,z) \stackrel{c}{\approx} \mathsf{MIM}_{\langle C,\mathsf{Ext}_{\mathsf{NMCom}}\rangle}^{\mathsf{Ext}}(m_1,z).$$

 $[CCG^+20]$ also considers the following *delay-input non-malleability* notion. A three-round non-malleable commitment scheme is delayed-input if the committer can choose the message to commit after the execution of the first two rounds. In particular, they consider the adaptive security that the adversary, after the two rounds, may adaptively chooses a message val and ask the honest sender in the left session to commit to this message in the third round. Now, let $\mathsf{MIM}^0_{\langle C,R\rangle}(z)$ stand for, in the above experiment, the joint distribution of the adversary's view and the value val' that he commits in the third round. On the other hand, consider the experiment where the honest sender on the left always commits to a random string in the last round. Let $\mathsf{MIM}^1_{\langle C,R\rangle}(z)$ stand for the joint distribution in this experiment.

Definition 14. A three-round commitment scheme is said to be delayed-input non-malleable if

$$\mathsf{MIM}^{0}_{\langle C,R\rangle}(z) \stackrel{\sim}{\approx} \mathsf{MIM}^{1}_{\langle C,R\rangle}(z).$$

Given these two definitions, $[CCG^+20]$ defines their non-malleability notion.

Definition 15 (Special Non-malleability). A three-round commitment scheme is special non-malleable if it satisfies

- It is non-malleable with respect to extraction.
- It is delayed-input non-malleable.
- It satisfies that the last message is pseudorandom. For any message m that the sender is to commit, the third-round message is indistinguishable from a uniformly random string given the first two rounds.

 $[CCG^+20]$ showed the construction of [GPR16] satisfies their definition.

Theorem 6 ([GPR16]). Assuming non-interactive commitments, there is a three-round commitment scheme with the special non-malleable property.

Reusability. To use it in our protocol, we need a reusable special non-malleability. That is, in both the non-malleable with respect to extraction and delayed-input non-malleable experiment, the MIM attacker is allowed to ask for multiple last round messages (committing to independent random strings) from the sender in the left session before he would finally ask for a challenge commitment from the sender on the left and trying to break the non-malleability.

We note that [GPR16] already satisfies this stronger notion due to its last message pseudorandomness property. In fact, their last message pseudorandomness property holds even if the receiver receives multiple third-round messages committing to arbitrary messages. Therefore, by a standard hybrid argument, we may first switch the (additional) multiple third-round messages that the MIM attacker gets to see to a purely random string. Afterward, the reusable special non-malleable property reduces to the non-reusable special non-malleable property as the MIM attacker could simulate those random strings himself.

7.1.3 Two-round (Reusable) Semi-Malicious MPC.

We also use a two-round reusable semi-malicious MPC in the plain model. In particular, this protocol should have the following properties.

- In the first round, parties send a message based only on their input.
- In the second round, given any function f, parties send a second-round message based on their secret state and function f.
- **Reusability.** The last round can be executed an unbounded number of times for different functions. However, every party's input is always fixed.

Such a two-round semi-malicious reusable MPC protocol has been constructed recently based on various assumption such as DDH assumption [BGMM20], SXDH assumption on asymmetric bilinear maps [BL20], LWE assumption [AJJM20, AJJM21, BJKL21], and the LPN assumption [BGSZ22].

7.2 Our Protocol

Our protocol uses the following building blocks.

- (rREcom₁, rREcom₂, rREcom₃): This is a reusable rewind-secure extractable commitment scheme defined in Section 7.1.1.
- (Ncom₁, Ncom₂, Ncom₃): This is the non-malleable commitment scheme defined in Section 7.1.2.
- $(rMPC_1, rMPC_2)$: This the two-round semi-malicious MPC protocol in Section 7.1.3.
- (TDGen₁, TDGen₂, TDGen₃): This is a *one-rewind secure* trapdoor generation protocol defined in Section 3.7.¹⁵

 $^{^{15}}$ We note that the specific trapdoor generation protocol construction [CCG⁺20] (based on the signature scheme) satisfies the *unique last round message* property. That is, given the first two rounds of the protocol, there is a unique last-round message that is accepting. In terms of reusing the protocol, this means that the sender in the trapdoor generation protocol will always send the same message in the third message of every reuse session.

- (ZAP₁, ZAP₂): The is the ZAP protocol. Recall that ZAP is both delayed-input and also unbounded rewind-secure as discussed in Section 3.6.
- $(rOT_1, rOT_2, rOT_3, rOT_4)$: This is the reusable OT protocol we constructed in Section 5.
- (Garble, Eval): This is a garbling scheme defined in Section 3.4.

High-level Summary. Figure 10 presents a high-level sketch of the protocol. On a basic level, parties shall execute the 2-round semi-malicious MPC in the third and fourth rounds to compute any function f. To ensure honest behavior, however, the following blocks are also added. First, parties shall commit to their input using the reusable delayed-input commitment scheme rREcom. Second, a three-round trapdoor generation protocol TDGen and a three-round non-malleable commitment scheme are executed in the first three rounds. Now, parties are supposed to prove their honest behavior in the third round by the first instance of the ZAP protocol $zap_{1,a}$, $zap_{2,a}$.¹⁶ However, this proof is not sent in the clear but used as choice bits for another OT protocol. Since the proof $zap_{2,a}$ is not sent in the clear, parties cannot verify the honesty of other parties before the fourth round. Therefore, parties cannot send their message rMPC₂ after it verifies all the $zap_{2,a}$ proofs. The labels of this garbled circuit are used as the sender's message in the OT protocol. Finally, another instance of the ZAP protocol is executed to prove the honest behavior of the last round. We refer the readers to the technical overview of [CCG⁺20] for a more detailed overview.

A formal description of our protocol is in Figure 11.



Figure 10: A pictorial view of the messages exchanged between party i and j. Every row corresponds to one round. The red part indicates where we make appropriate modifications to $[CCG^+20]$. The OT protocol is instantiated with a reusable OT protocol; the (bounded) rewind-secure WI proofs are instantiated with ZAP; the MPC is instantiated with a two-round reusable MPC.

Intuition for Reusability. $[CCG^+20]$ proves that their protocol is secure in the standalone setting. However, when one reuses the first two rounds of the protocol, this might cause issues as each reuse session shares the same first two rounds. In particular, in the security proof, when we try to reduce the indistinguishability between hybrids to the security of a particular building block, we might rely on the external challenger to send us the messages belonging to that building block for *all reuse sessions*. Therefore, it is crucial that all building blocks individually are reusable. And, indeed, as long as each building blocks are reusable, one can use the same proof as $[CCG^+20]$ to prove the reuse security. In particular, the indistinguishability between hybrids shall reduce to the *reusable* security of specific building blocks.

Theorem 7. Assuming the reusable security of the building blocks, the protocol in Figure 11 is a four-round reusable MPC protocol.

For ease of presentation, we omit the (\cdot) superscript indicating the reuse sessions.

 $^{^{16}}$ ZAP proves that either the party is generating all the messages correctly, or the non-malleable commitment commits to a valid trapdoor.

• Language $\mathcal{L}_a^{i \to j}$. Given the first two rounds, a statement

$$\mathsf{st}_a^{i \to j} \coloneqq \left(\mathsf{rmpc}_{i,1}, \left\{\mathsf{rrecom}_3^{i \to k}\right\}_k, \mathsf{ncom}_3^{i \to j}\right)$$

is in the language $\mathcal{L}_a^{i \to j}$ with witness

$$w_a^{i \to j} := \left(x_i, r_i, \left\{ r_{\text{rrecom}}^{i \to k} \right\}_k, r_{\text{ncom}}^{i \to j}, t \right)$$

if either one of the following is true.

- Honest witness. (1) rmpc_{*i*,1} is a honestly generated with input x_i and randomness r_i ; (2) For all k, (rrecom^{*i*→*k*}, rrecom^{*k*→*i*}, rrecom^{*i*→*k*}) is an honest commitment of (x_i, r_i) using randomness $r^{i\to k}_{\text{rrecom}}$.
- **Trapdoor witness.** $(\mathsf{ncom}_1^{i \to j}, \mathsf{ncom}_2^{i \to j}, \mathsf{ncom}_3^{i \to j})$ is an honest commitment of t with randomness $r_{\mathsf{ncom}}^{i \to j}$ such that t is a valid trapdoor with respect to $\mathsf{td}_{1,j}$.
- Language $\mathcal{L}_{b}^{i \to j}$. Given the first three rounds, a statement

$$\mathsf{st}_b^{i \to j} \coloneqq \left(\left\{ \mathsf{rot}_4^{i \to k} \right\}_k, \widehat{C_i}, \mathsf{ncom}_3^{i \to j} \right)$$

is in the language $\mathcal{L}_{b}^{i \to j}$ with witness

$$w_b^{i \rightarrow j} := \left(x_i, r_i, \mathsf{rmpc}_{2,i}, \left\{ r_{\mathsf{rrecom}}^{i \rightarrow k}, r_{\mathsf{rot}}^{i \rightarrow k} \right\}_k, r_{\mathsf{gc},i}, r_{\mathsf{ncom}}^{i \rightarrow j}, t \right)$$

if either one of the following is true.

- Honest witness. (1) $(\mathsf{rmpc}_{i,1},\mathsf{rmpc}_{i,2})$ is honestly generated with input x_i and randomness r_i ; (2) for all k, $(\mathsf{rrecom}_1^{i \to k}, \mathsf{rrecom}_2^{k \to i}, \mathsf{rrecom}_3^{i \to k})$ is an honest commitment of (x_i, r_i) using randomness $r_{\mathsf{rrecom}}^{i \to k}$; (3) $(\widehat{C}_i, \{\mathsf{lab}_{w,b}\}_{w,b})$ is an honest garbling of C_i with randomness $r_{\mathsf{gc},i}$; (4) for all k, $\mathsf{rot}_4^{i \to k}$ is honestly generated with randomness $r_{\mathsf{rot}}^{i \to k}$ and messages $\{\mathsf{lab}_{w,b}\}_{w \in \mathsf{P}_k, b}$.
- **Trapdoor witness.** $(\mathsf{ncom}_1^{i \to j}, \mathsf{ncom}_2^{i \to j}, \mathsf{ncom}_3^{i \to j})$ is an honest commitment of t with randomness $r_{\mathsf{ncom}}^{i \to j}$ such that t is a valid trapdoor with respect to $\mathsf{td}_{1,j}$.
- **Circuit.** The circuit C_i does the following.
 - Hardwired Inputs. $\operatorname{rmpc}_{2,i}, \left\{\operatorname{st}_{a}^{j \to i}\right\}_{j}, \text{ and } \left\{\operatorname{zap}_{1,a}^{i \to j}\right\}_{j}$
 - Inputs. $\left\{ \mathsf{zap}_{2,a}^{j \to i} \right\}_{i}$
 - Computation. If for all j, $(zap_{1,a}^{i \to j}, zap_{2,a}^{j \to i})$ is a valid proof of the statement $st_a^{j \to i}$, output rmpc_{2,i}. Otherwise, output \perp .
- Protocol Description.
 - 1. P_i computes/broadcasts the following:
 - Trapdoor generation protocol: $\mathsf{td}_{1,i} \leftarrow \mathsf{TDGen}_1(r_{\mathsf{td},i})$

For every $j \neq i$,

- All Commitments:

*
$$\operatorname{rrecom}_{1}^{i \to j} \leftarrow \operatorname{rREcom}_{1}(r_{\operatorname{rrecom}}^{i \to j})$$

- * $\operatorname{ncom}_{1}^{i \to j} \leftarrow \operatorname{Ncom}_{1}(r_{\operatorname{ncom}}^{i \to j})$
- Reusable OT: $\mathsf{rot}_1^{i \to j} \leftarrow \mathsf{rOT}_1(r_{\mathsf{rot}}^{i \to j})$
2. P_i computes/broadcasts the following. For every $j \neq i$:

- Trapdoor generation protocol: $\mathsf{td}_2^{i \to j} \leftarrow \mathsf{TDGen}_2$
- $\text{ ZAP: } \mathsf{zap}_{1,a}^{i \to j} \leftarrow \mathsf{ZAP}_1, \, \mathsf{zap}_{1,b}^{i \to j} \leftarrow \mathsf{ZAP}_1$
- All Commitments:
 - $* \operatorname{rrecom}_2^{i o j} \leftarrow \operatorname{rREcom}_2$
 - * $\mathsf{ncom}_2^{i \to j} \leftarrow \mathsf{Ncom}_2$
- Reusable OT: $\mathsf{rot}_2^{i \to j} \leftarrow \mathsf{rOT}_2$

For the k^{th} reuse session, computes the following.

- 3. P_i computes/broadcasts the following.
 - Reusable MPC: $\mathsf{rmpc}_{1i} \leftarrow \mathsf{rMPC}_1(x_i, r_i)$. Here, x_i is P_i 's input.
 - Let $\mathsf{td}_{2,i}$ be the concatenation of $\mathsf{td}_2^{j \to i}$ for all $j \neq i$. Trapdoor generation protocol: $\mathsf{td}_{3,i} \leftarrow \mathsf{TDGen}_3(\mathsf{td}_{1,i}, \mathsf{td}_{2,i}, r_{\mathsf{td},i})$.

For every $j \neq i$:

- All commitments.
 - * $\operatorname{rrecom}_{3}^{i \to j} \leftarrow \operatorname{rREcom}_{3}((x_{i}, r_{i}), r_{\operatorname{rrecom}})$ commits to input x_{i} and randomness r_{i} used for the semi-malicious protocol.
 - * $\operatorname{ncom}_{3}^{i \to j} \leftarrow \operatorname{Ncom}_{3}(t, r_{\operatorname{ncom}})$ commits to a random string t. (Note that parties keep committing to a new string t for each reuse session until the first reuse session where the third round messages do not cause an abort. In this case, this non-malleable commitment is fixed and shall be sent in all future reuse session as the third round message of the non-malleable commitment protocol.)
- Sample $zap_{2,a}^{i \to j} \leftarrow ZAP_2$ such that $(zap_{1,a}^{j \to i}, zap_{2,a}^{i \to j})$ proves the statement $st_a^{i \to j}$ using the honest witness. Note that $zap_{2,a}^{i \to j}$ will not be sent.
- $-\operatorname{rot}_{3}^{i \to j} \leftarrow \operatorname{rOT}_{3}(\operatorname{zap}_{2,a}^{i \to j}, r_{\operatorname{rot}}^{i \to j})$. Here, P_{i} is the receiver in the OT protocol with choice bits $\operatorname{zap}_{2,a}^{i \to j}$.
- 4. P_i computes/broadcasts the following.
 - Check Trapdoor validity: if there exists j such that $(\mathsf{td}_{1,j}, \mathsf{td}_{2,j}, \mathsf{td}_{3,j})$ is invalid, abort.
 - Reusable rewind-secure MPC: $\mathsf{rmpc}_{2,i} \leftarrow \mathsf{rMPC}_2(x_i, r_i)$. Note that $\mathsf{rmpc}_{2,i}$ will not be sent.
 - Garbled Circuits: \widehat{C}_i , where $(\widehat{C}_i, \{\mathsf{lab}_{w,b}\}_{w,b}) \leftarrow \mathsf{Garble}(C_i, r_{\mathsf{gc},i}).$

For every $j \neq i$:

- OT messages: $\operatorname{rot}_{4}^{i \to j} \leftarrow \operatorname{rOT}_{4}({\operatorname{\mathsf{lab}}_{w,b}}_{w \in \mathsf{P}_{j,b}}, r_{\mathsf{rot}}^{i \to j})$. That is, P_{i} is the sender in the OT protocol with party j's input labels as its messages.
- In order to facilitate other parties to recover the input labels corresponding to P_i for the garbled circuit \widehat{C}_j , P_i shall send its secret state to other parties.

However, instead of broadcasting $r_{\text{rot}}^{i \to j}$ as did in [CCG⁺20], parties shall broadcast the secret-key for decrypting the ciphertext of the last round (refer to Remark 3).

- This change is necessary as the rOT is no longer reusable if the entire secret state $r_{rot}^{i \rightarrow j}$ is revealed.
- Sample $zap_{2,b}^{i \to j} \leftarrow ZAP_2$ such that $(zap_{1,b}^{j \to i}, zap_{2,b}^{i \to j})$ proves the statement $st_b^{i \to j}$ using the honest witness.
- Output Computation. Party P_i computes the following.
 - Verify that $(\mathsf{zap}_{1\,h}^{j \to i}, \mathsf{zap}_{2\,h}^{i \to j})$ is a valid proof of the statement $\mathsf{st}_{h}^{i \to j}$.

- Extract the OT messages: for all $j \neq i$, and for all $k \neq j$, extract $\{\mathsf{lab}_{w,b}\}_{w \in \mathsf{P}_k}$ from $\mathsf{rot}_4^{j \to k}$. This is feasible since P_k broadcasts the decryption key for the rOT messages it receives.
- Evaluate the garbled circuits: for all $j \neq i$, $\mathsf{rmpc}_{2,j} = \mathsf{Eval}(\widehat{C_j}, \{\mathsf{lab}_{w,b}\}_w)$.
- Given all the messages $\{\mathsf{rmpc}_{i,j}\}_{i\in[2],j\in[n]}$ of the semi-malicious MPC protocol, evaluate the output of the protocol.

Figure 11: A formal description of our reusable MPC protocol

7.3 The Simulator

Our simulator is similar to $[CCG^+20]$. We formally state it below.

Step 1. Check Adversary Abort. In this step, the simulator checks if the adversary aborts all reuse sessions or not.

- The simulator follows the honest party's protocol using input 0.
- It runs all the reuse sessions until there is one reuse session where the adversary does not abort the third round.
- Check Abort. If the adversary aborts in all reuse sessions, the simulator aborts and outputs the view of the adversary. Otherwise, we say the check abort step succeeds. Let us denote this reuse session where the adversary sends an accepting third-round message for the first time by u.
- Check Implicit Abort. If the adversary did not abort explicitly, the simulator now runs a lookahead thread to extract the proof $zap_{1,a}$ from the reusable OT protocol. As long as there is one malicious party whose proof does not verify, we call this implicit abort.

Remark 5. Recall in our reusable OT protocol, two seeds $r_0^{(i)}, r_1^{(i)}$ are committed using the delayed-input extractable commitment scheme in the third round. When one extracts from the rOT protocol, one extracts the key committed in the first round of the delayed-input extractable commitment scheme, which, in turn, allows extraction of $r_0^{(i)}, r_1^{(i)}$ and further allows extraction of the choice bits.

Moreover, the extraction of the key allows the extraction of choice bits for all future third-round rOT messages (as they all share the fixed first-round message). Therefore, this extraction only needs to be run once.

Step 2. Rewinding. The simulator runs a set of lookahead threads as follows.

- The simulator rewinds the protocol to the completion of the first round and runs a number of lookahead threads.
- In each lookahead thread, the simulator uses the honest party's strategy using input 0. It runs until this thread satisfies that (1) the first u 1 reuse sessions aborts and (2) the third-round message of the *u*-th reuse session is accepting.
- No Abort Case.
 - We say a lookahead thread is GOOD (with respect to some honest party *i*) if, in the *u*-th reuse session, all malicious parties' third-round message is accepting and the proofs $zap_{1,a}$ extracted from their rOT message (to party *i*) verify.
 - The simulator runs as many lookahead threads as needed to get 12λ GOOD threads.
- Implicit Abort Case.

- We say a look ahead thread is $\mathsf{IMPLICIT}$ if, in the u-th session, all malicious parties' thirdround message is accepting.
- The simulator runs as many look ahead threads as needed to get 12λ IMPLICIT threads.

Step 3. Extraction. In this step, the simulator extracts the trapdoors and inputs.

- Without loss of generality, assume $12\lambda > 5n$.
- No abort case. Since there are > 5n GOOD threads, there exists an honest party i^* such that there exist 5 GOOD threads with respect to i^* . Using these 5 GOOD threads to extract the trapdoor and input from all adversaries. Note that 5 accepting transcripts are sufficient for extracting both the inputs and trapdoors.
- Implicit abort case. Similar to no abort case, except that only trapdoors are extracted.

Step 4. Estimating abort probability. Recall that T denotes the number of lookahead threads sampled in order to get 12λ GOOD or IMPLICIT threads. Set $\varepsilon' = 12\lambda/T$ to be the estimation of abort probability.

Step 5. Re-sampling the main thread. The simulator now re-samples the main thread until the *u*-th reuse session's third round. That is, in the first u - 1 reuse session, the adversary aborts in the third round. In the *u*-th session, the adversary's third-round messages are accepting.

- It uses the honest party's strategy with input 0 in the first u 1 reuse sessions.
- In the third round of the *u*-th reuse session, it does the following
 - The non-malleable commitment is committed to the extracted trapdoor.
 - The proof $\mathsf{zap}_{1,a}$ is generated using the trapdoor witness.
 - The rREcom commits to 0 as its input.
 - For the first message of the MPC, sample it using the simulator of the semi-malicious MPC.
- Abort and re-sample condition.
 - In the no abort case, it ensures that in the first u-1 reuse sessions, the adversary all aborts in the third round. Only in the third round of the u-th reuse session, the adversary does not abort and the extracted proof $zap_{1,2}$ also verifies.
 - In the implicitly abort case, it ensures that in the first u 1 reuse sessions, the adversary all aborts in the third round. In the third round of the reuse session, the adversary does not abort and the extracted proof $zap_{1,2}$ does not verify.
 - It runs for at most $\min(2^{\lambda}, \lambda^2/\varepsilon')$ times.

Step 6. Finishing the main thread. The simulator now finishes the main thread. This includes not only the fourth round of *u*-th reuse session but *all future reuse sessions*.

- For the fourth round of the current freezed reuse session:
 - No abort case.
 - * Query the ideal functionality and get the output.
 - * Invoke the simulator of the semi-malicious MPC with the obtained output and generate the simulated second-round message $\mathsf{rmpc}_{2,i}$.
 - * Invoke the simulator of the garbled circuit with $\mathsf{rmpc}_{2,i}$ as the output.
 - * The fourth round of the OT message is sampled using the simulated labels as input

messages.

- * The proof $zap_{b,2}$ is sampled using the trapdoor witness.
- * It checks the fourth-round message sent by the adversary, if they are accepting, instruct the ideal functionality to send the output to the honest party. Otherwise, instruct the ideal functionality to send \perp to the honest parties.
- Implicit abort case.
 - * Invoke the simulator of the garbled circuit with \perp as the output.
 - * The fourth round of the OT message is sampled using the simulated labels as input messages.
 - * The proof $zap_{b,2}$ is sampled using the trapdoor witness.
 - * Instruct the ideal functionality to send \perp to the honest party.
- For all future reuse sessions:
 - The third round is sampled similarly in step 5. That is,
 - * The non-malleable commitment is sampled using the trapdoor.
 - * The proof $zap_{1,a}$ is sampled using the trapdoor witness.
 - * The rREcom commits to 0.
 - * The first round message of the MPC is sampled using the simulator.
 - * **Determine abort case.** Based on the adversary's third round message, we label this session either as explicit abort, implicit abort, or no abort.
 - The fourth round is sampled similarly as above. If the adversary explicitly aborts, no message needs to be sent. If the adversary either implicitly aborts or does not abort, we use the strategy described above.

Running time of the simulator. Following a similar analysis as in [CCG⁺20], we remark that the simulator shall run in the expected polynomial time. Fix u to be the first reuse session that the third round is accepting. Assume that the simulator goes into step 2 with probability ε . The only steps where the adversary might run in exponential time are

- Step 2, where the simulator samples 12λ lookahead threads for extraction. The expected total number of threads of this is $12\lambda/\varepsilon$.
- Step 5. By similar analysis in [BGJ⁺18], if the probability estimation is correct, this step ends in λ^2/ε steps. However, there is a $2^{-\lambda}$ probability that the probability estimation is wrong, in which case, the simulator will end in at most 2^{λ} steps.

Overall, for a fixed u, the total expected running time is

$$\mathsf{poly}(\lambda) + \mathsf{poly}(\lambda) \cdot \varepsilon \cdot (12\lambda/\varepsilon + (1 - 2^{-\lambda}) \cdot \lambda^2/\varepsilon + 2^{-\lambda} \cdot 2^{\lambda}) = \mathsf{poly}(\lambda).$$

Since there are at most polynomially many reuse sessions, the overall running time is again expected polynomial.

7.4 The Hybrids

Assume that there is an adversary that can distinguish the real and ideal world by a non-negligible advantage. Recall that u is the first reuse session such that the check abort step succeeds at the u-th session. Since there are at most polynomial reuse sessions, there must exist a u such that the adversary can distinguish the real and ideal world conditioned on u by a non-negligible advantage μ .

 $\mathsf{Hyb}_{\mathsf{REAL}}$. This is the real world.

Hyb_0 . Determine Abort in the 3rd round and extraction.

- The simulator executes the protocol using the honest party's strategy. If the adversary aborts in every third-round message across all reuse sessions, the simulator outputs the view of the adversary and aborts.
- If "check abort" succeeds (i.e., there is one reuse session where the adversary does not abort in the third-round message.), the simulator checks if there is an implicit abort by extracting $zap_{2,a}$ from the rOT protocol.
- In either no-abort or implicit-abort cases, the simulator rewinds the execution to after the completion of the first round. It generates $\frac{5n\lambda}{\mu}$ lookahead threads. In each lookahead, it uses the honest parties' strategy and inputs.
- In the case of implicit-abort, the simulator extracts the trapdoors and proofs. In the cases of no-abort, the simulator extracts the inputs, trapdoors, and proofs.
- The simulator outputs $\perp_{extract}$ if the extraction fails in either case.
- The simulator now re-samples the main thread from the completion of round 1. It rewinds and resamples the main thread for at most $\frac{\lambda}{\mu}$ times for this step.

Since we assume μ is non-negligible, this hybrid will end in polynomial time.

 Hyb_1 . Using input 0 in the aborting step. In this hybrid, the "check abort" is done using the fake input 0. We switch to this hybrid in a sequence of hybrids as follows.

- Hyb_{1,0}. Switch the rOT choice bits to 0. That is, for all honest parties, the simulator sends $rOT_3^{i\to j}(0)$ instead of $rOT_3^{i\to j}(zap_2^{i\to j})$.
- Hyb_{1,1}. Switch the rREcom to 0. That is, the input committed in rREcom is switched from honest input to 0.
- $Hyb_{1,2}$. Switch the input of Π to 0. That is, we switch the input to the two-round semi-malicious MPC protocol Π from honest input to 0.
- Hyb_{1,3}. Switch the rOT choice bits to zap^{i→j}_{2,a}. We finally switch the choice bits in the OT protocol from 0 to the honest proof zap^{i→j}_{2,a}.

Hyb₂. Using input 0 in the lookahead thread. The simulator does this switch one lookahead thread at a time. In each lookahead thread, the simulator switches things in a sequence of hybrids as follows.

We note that within each lookahead thread, we again switch things one reuse session at a time. For every k, the following changes are only made to the k-th lookahead thread.

- $Hyb_{2,k,0}$. Switch the Ncom. In the k-th lookahead thread, for the non-malleable commitment, the simulator, instead of committing to a random string, commits to the trapdoor *extracted from other reuse sessions*.
- $Hyb_{2,k,1}$. Switch the ZAP proof. In the k-th lookahead thread, the simulator samples $zap_{2,a}$ using the trapdoor witness (instead of the honest witness).
- $Hyb_{2,k,2}$. Switch the rEcom. In the k-th lookahead thread, the simulator samples rEcom₃ commits to the input 0 (instead of the honest input).
- $Hyb_{2,k,3}$. Switch the input of Π to 0. In the k-th lookahead thread, the simulator generates the first round message of Π using input 0 (instead of the honest input).
- $Hyb_{2,k,4}$. Switch the ZAP proof. In the k-th lookahead thread, the simulator now samples $zap_{2,a}$ back to using the honest witness.

• $Hyb_{2,k,5}$. Switch the Ncom. In the *k*-th lookahead thread, for the non-malleable commitment, the simulator switches back to committing to a random string.

Additionally, we shall also switch the input from honest to 0 for the first u - 1 reuse sessions of the main thread.

Hyb₃. Switch Ncom to commit to the extracted trapdoor in the main thread. As described in the simulator, if the check abort step succeeds at the *u*-th reuse session. Our re-sampled main thread will also maintain that the *u*-th session is the first session, where the adversary does not abort. We shall switch the Ncom to trapdoor for all reuse sessions starting from u. (Recall that once a reuse session does not abort in the third round, the non-malleable commitment is fixed for all reuse sessions in the future.)

 Hyb_4 . Switch the proof $zap_{a,2}$ to using the trapdoor witness in the main thread. This is done one reuse session at a time starting from the *u*-th reuse session.

 Hyb_5 . Switch the proof $zap_{b,2}$ to using the trapdoor witness in the main thread. This is again done one reuse session at a time starting from the *u*-th reuse session.

 Hyb_6 . Switch rREcom to commit to 0 in the main thread. This is done one reuse session at a time starting from the *u*-th reuse session.

Hyb₇. Simulate Π in the main thread. This is done one reuse session at a time starting from the *u*-th reuse session. The simulator queries the ideal functionality and obtains the output of the protocol. Using this output, it simulates the rMPC messages.

Hyb₈. Simulate the Garbled Circuit in the main thread. This is done one reuse session at a time starting from the *u*-th reuse session. Depending on implicit abort or no abort, the simulator for the garbled circuit is invoked with output \perp or rmpc_{2,i}.

Hyb_{IDEAL}. Run the actual probability estimation. In this hybrid, the simulator does not sample only $5n\lambda/\mu$ threads but as many as needed to get 12λ GOOD thread. It also estimate the abort probability $\varepsilon' = 12\lambda/T$ and continues to re-sample the main thread at most $\min(2^{\lambda}, \lambda^2/\varepsilon')$ times.

7.5 Indistinguishability of Hybrids.

Our proof follows the same proof structure as in $[CCG^+20]$. We shall highlight if there is a difference in the proof due to reuse.

As in $[CCG^+20]$, we maintain the following invariant.

Definition 16 (Invariant). We say Event occurs if there exists an i, j, k such that

- P_i is an honest party, P_j is a malicious party, and k is a reuse session.
- $\operatorname{Ext}_{\operatorname{NMCom}}$ from the non-malleable commitment transcript $\operatorname{ncom}_1^{j \to i}, \operatorname{ncom}_2^{i \to j}, \operatorname{ncom}_3^{j \to i, (k)}$ outputs t_i .
- TDValid $(td_{1,i}, t_i) = 1$.

That is, Event happens if the extractor outputs a valid trapdoor t_i from the non-malleable commitment from a malicious party P_i to an honest party P_i in some reuse session. The invariant we maintain is

 $\Pr[\mathsf{Event}] \leq \mathsf{negl}(\lambda).$

Claim 23. The invariant holds in the hybrid Hyb_{REAL}.

Proof. The proof of $[CCG^+20]$ shows that if Event happens with non-negligible probability, one can break the 1-rewind-security of the trapdoor. In particular, the adversary interacts with the external challenger for the trapdoor generation protocol. It simulates the hybrids by simulating all the messages except for the trapdoor generation protocol between *i* and *j*. Here, $td_{1,i}$ is set to be td_1 it receives from the external challenger. It forwards $td_{2,i}$ to the external challenger as td_2 and receives back td_3 , which is set to be $td_{3,i}$. It then freezes the first round of the hybrid and simulates another thread. Again, it simulates all the messages except for the extractor Ext_{NMCom} to extract the message committed by Party *j*. If Event happens with non-negligible probability, Ext_{NMCom} will output a valid trapdoor with non-negligible probability and, hence, breaking the 1-rewind-security of the trapdoor generation protocol.

In the reuse setting, we need to simulate all reuse sessions when we simulate the hybrids. However, since the trapdoor generation protocol has a fixed third-round message across all reuse sessions, receiving td_3 from the external challenger enables us to simulate all reuse sessions.

Claim 24. The invariant holds in Hyb_0 .

Proof. Since there is no change in the main thread. The invariant continues to hold. \Box

Claim 25. Hyb₀ is indistinguishable from Hyb_{REAL} with probability $\leq \frac{\mu}{4} + \operatorname{negl}(\lambda)$.

Proof. The proof is similar to [CCG⁺20]. If the probability of "not abort" in the check abort step is $< \mu/4$, then the claim is correct as there is no difference when the adversary aborts.

If the adversary does not cause an abort with probability $\geq \mu/4$, then by Chernoff bound, with $1 - \operatorname{negl}(\lambda)$ probability, there will be 5 GOOD thread with respect to some honest party i^* out of $\frac{5n\lambda}{\mu}$ threads. Additionally, given 5 GOOD threads, the extraction succeeds with overwhelming probability.

Finally, out of λ/μ thread, we will successfully re-sample the main thread with overwhelming probability. Hence, the claim is correct.

In the rest of the proof, we only focus on the no-abort case. Implicit abort can be handled similarly.

Claim 26. The invariant holds in $Hyb_{1,0}$.

Proof. Since there is no difference in the main thread, the invariant still holds.

Claim 27. Assuming the hiding property of the reusable OT against malicious senders, $Hyb_{1,0}$ is indistinguishable from Hyb_0 .

Proof. Similar to $[CCG^+20]$, we shall prove that switching the OT protocol between every pair of parties (i, j) one at a time is indistinguishable. Since there are at most n^2 such pairs, one concludes that $Hyb_{1,0}$ is indistinguishable from Hyb_0 .

Note that, in our case, there are multiple reuse sessions and we rely on the external challenger to generate all the rOT messages in all reuse sessions. However, since the receiver's privacy of rOT holds in the reuse setting. The distinguishability of the two hybrids still results in a successful attack on the receiver's privacy in rOT.

Consider the adversary that interacts with an external challenger for the rOT protocol. The adversary simulates the protocol for all messages except for the rOT protocol between the receiver Party i and the sender Party j. It forwards the message it receives from the external challenger as the receiver's message for party i and forwards the message it receives from Party j as the malicious sender's message to the external challenger. For every reuse session, it gives the external challenger $zap_{2,a}^{i \to j}$ and 0. The external challenger either always uses the input $zap_{2,a}^{i \to j}$ or always uses 0. Therefore, if the two hybrids are distinguishable, we break the receiver's privacy of the reusable OT protocol.

Claim 28. The invariant holds in $Hyb_{1,1}$.

Proof. Since there is no difference in the main thread, the invariant still holds.

Claim 29. Assuming the hiding property of the rREcom, $Hyb_{1,1}$ is indistinguishable from $Hyb_{1,0}$.

Proof. If the two hybrids are distinguishable, we break the reusable privacy of rREcom. Note that since there are multiple reuse sessions, we rely on the external challenger to send all the third-round messages. Recall that in the third round, the sender sends $\{k \oplus p_j(0), p_j(z)\}_j$ (which is identical across all reuse sessions) and additionally, SKE.Enc $(k, m^{(i)})$ committing to some $m^{(i)}$ in the *i*-th session. Since k is hiding, by the CPA-security of the SKE scheme, we can switch all the inputs from honest to 0 without being detected. \Box

Claim 30. The invariant holds in $Hyb_{1,2}$.

Proof. Since there is no difference in the main thread, the invariant still holds.
$$\Box$$

Claim 31. Assuming the privacy of Π , $\mathsf{Hyb}_{1,2}$ is indistinguishable from $\mathsf{Hyb}_{1,1}$.

Proof. We switch the input from honest to 0 one reuse session at a time.

Note that we use a two-round semi-malicious protocol that is computed in the third round and fourth rounds. Hence, the instance of Π in each reuse session is completely independent of each other. Consequently, the reduction from the indistinguishability of switching the input of Π in one reuse session to the privacy of Π is straightforward. That is, we only rely on the external challenger to send us one message, which is the first round message of Π using either honest input or 0, and this enables us to simulate the entire hybrid. Hence, if $Hyb_{1,2}$ and $Hyb_{1,1}$ are distinguishable, we break the privacy of Π .

Claim 32. The invariant holds in $Hyb_{1,3}$.

Proof. Since there is no difference in the main thread, the invariant still holds.

Claim 33. Assuming the receiver privacy of the reusable OT, $Hyb_{1,3}$ and $Hyb_{1,2}$ are indistinguishable.

Proof. The proof is analogous to the proof of Claim 27.

As we go into switching things in the lookahead thread. We now argue that the invariant holds even in the lookahead thread as well. Initially, all the lookahead threads are the same as the main thread. Therefore, by Claim 23, the invariant holds in every lookahead thread.

Claim 34. Assuming the non-malleability with respect to the extraction property of Ncom, the invariant holds in the $Hyb_{2,k,0}$.

Proof. If the invariant does not hold, we shall construct an adversary that breaks the non-malleability with respect to extraction of the Ncom. In particular, the adversary \mathcal{A}_{Ncom} picks a random honest party *i*, malicious party *j*, and a reuse session *k*. It simulates all messages of the hybrid except for those in the non-malleable commitment protocol between *i* and *j*, which it receives from the external challenger. Note that our reusable non-malleable commitment allows the adversary to get multiple third-round messages committing to random messages before finally obtaining the challenge commitment. Therefore, the adversary gets third-round messages of the Ncom from all reuse sessions from the external challenger.

The adversary also creates 5 lookahead threads as the hybrid described. With non-negligible probability, those 5 lookahead threads are GOOD with respect to some honest party. Therefore, we extract the trapdoor from j with a non-negligible probability. The trapdoor t_j and a random message r are forwarded to the external challenger and the challenge commitment commits to either one of them.

Depending on which message the commitment commits to, we either simulate $Hyb_{2,k,0}$ or $Hyb_{2,k-1,5}$. Since the invariant holds in the previous hybrid and does not hold in this hybrid, if Ext_{NMCom} outputs a valid trapdoor, we must be in hybrid $Hyb_{2,k,0}$. This breaks the non-malleability with respect to extraction.

Claim 35. Assuming the hiding property of Ncom, $Hyb_{2,k,0}$ is indistinguishable from $Hyb_{2,k-1,5}$.

Proof. The changes are made in the lookahead thread. To prove indistinguishability, we shall prove that the extraction continues to succeed. We already established that the invariant holds in this hybrid. Therefore, if the proof $zap_{2,a}$ verifies, with overwhelming probability, the transcript is well-formed. Therefore, as long as we prove that the probability of $zap_{2,a}$ verifies only changes negligibly, the probability that we receive a well-formed transcript is almost the same, and hence, the extraction succeeds with almost the same probability.

Now, assume for contradiction that the probability of $zap_{2,a}$ verifying changes by a non-negligible amount, we shall break the hiding property of Ncom as follows. The adversary \mathcal{A}_{Ncom} picks a random honest party i,

malicious party j, and a reuse session k. It simulates all messages of the hybrid except for those in the nonmalleable commitment protocol between i and j, which it receives from the external challenger. Note that our reusable non-malleable commitment allows the adversary to get multiple third-round messages committing to random messages before finally obtaining the challenge commitment. Therefore, the adversary gets thirdround messages of the Ncom from all reuse sessions from the external challenger.

The adversary also creates 5 lookahead threads as the hybrid described. With non-negligible probability, those 5 lookahead threads are GOOD with respect to some honest party. Therefore, we extract the trapdoor t_j with a non-negligible probability. Now the adversary sends t_j and a random value r to the external challenger and receives back a challenge commitment. It then forwards this message to the adversary and receives back the third-round message of the reusable OT protocol, from which he can extract the proof $zap_{2,a}$. Now the adversary checks if the proof $zap_{2,a}$ verifies. If so, it guesses the committed message to be t_j ; otherwise, it guesses r. Since there is a non-negligible difference in the probability that $zap_{2,a}$ verifies, one distinguishes the commitment of t_j from r with a non-negligible advantage.

Claim 36. Assuming the witness indistinguishable property of the ZAP, the invariant continues to hold in $Hyb_{2,k,1}$.

Proof. We know the invariant holds in the previous hybrid. If the invariant does not hold in this hybrid, we may break the witness indistinguishability of ZAP as follows.

The adversary \mathcal{A}_{zap} shall simulate all messages in the hybrid except for the ZAP proof that we are switching (i.e., a single instance of the ZAP protocol). Observe that although the ZAP instances in different reuse sessions might share the same first-round message, we do not rely on the external challenger to generate those proofs for us since the prover has no secret state.

Now, the adversary also creates 5 lookahead threads, which is GOOD with respect to some party with a non-negligible probability. From these lookahead threads, it extracts the trapdoor t_j from party j. It then forwards the statement and two witnesses (the trapdoor witness and the honest witness) to the external challenger and receives back proof using one of them.

Now, \mathcal{A}_{zap} runs the extractor $\mathsf{Ext}_{\mathsf{NMCom}}$ and extracts the trapdoor from the non-malleable commitment from the adversary and checks if it is valid. If it is valid, it guesses that the witness is the trapdoor witness. Otherwise, it guesses the witness to be the honest witness. Since there is a non-negligible change in the probability if the extracted value is a valid trapdoor or not, this breaks the witness indistinguishability of the ZAP protocol.

Claim 37. Assuming the witness indistinguishable property of the ZAP, $Hyb_{2,k,1}$ is indistinguishable from $Hyb_{2,k,0}$.

Proof. The changes are in the lookahead thread; we need to guarantee that the extraction succeeds with roughly the same probability. Since we have already established that the invariant holds, it must be the case that the transcript is well-formed if the proof verifies. Therefore, if the two hybrids are distinguishable, it must be the case that the probability that some of those proofs verify changes non-negligibly. We shall use this non-negligible difference to distinguish which witness the external challenger uses. The rest of the proof is similar to the proof of Claim 35 and Claim 36.

Claim 38. The invariant holds in $Hyb_{2,k,2}$.

Proof. This is done by a sequence of hybrids.

• In this hybrid, we switch the non-interactive commitment Com in the construction of rREcom from committing to p_i to junk values \perp .

Claim 39. The invariant holds in this hybrid.

Proof. We switch the Com one at a time and show that if the invariant does not hold, then we break the hiding property of Com.

Specifically, the adversary \mathcal{A}_{Com} sends the external challenger p_i and \perp , and receives back a commitment c. It simulates the hybrids as described except for the commitment (that commits to either p_i or \perp).

It then runs several lookahead threads and invokes $\mathsf{Ext}_{\mathsf{NMCom}}$ to extract the committed value t. If t is a valid trapdoor, it outputs \bot ; otherwise, it outputs p_i .

Since the invariant holds in the previous hybrid, if it does not hold in this hybrid, the adversary breaks the hiding property of Com with a significant advantage. \Box

• In this hybrid, instead of sending $\{p_i(0) \oplus k, p_i(z_i)\}_i$, SKE.Enc(k, input) as the third round message of rREcom, the simulator sends $\{U_i, p_i(z_i)\}_i$, SKE.Enc(k, input), where U_i is uniformly random.

Claim 40. The invariant holds in this hybrid.

Proof. Note that we only need 3 lookahead threads to extract the non-malleable commitment using $\mathsf{Ext}_{\mathsf{NMCom}}$ and the trapdoor. Since our polynomial is of degree B = 4, getting 3 evaluations of the degree-4 polynomial, the evaluation $p_i(0)$ is still uniformly random. Hence, if the invariant holds in the previous hybrid, the invariant must also hold in this hybrid.

• In this hybrid, we switch the input to 0. That is, in the last round of rREcom, we send $\{U_i, p_i(z_i)\}_i$, SKE.Enc(k, 0).

Claim 41. The invariant holds in this hybrid.

Proof. This is due to the CPA security of the SKE scheme. The proof is similar to the proof of Claim 39.

• In this hybrid, we switch the third round of the commitment back to $\{p_i(0) \oplus k, p_i(z_i)\}_i$, SKE.Enc(k, 0).

Claim 42. The invariant holds in this hybrid.

Proof. The proof is identical to the proof of Claim 40.

• In this hybrid, we switch the non-interactive commitment Com in the construction of rREcom from committing to junk \perp to p_i .

Claim 43. The invariant holds in this hybrid.

Proof. The proof is identical to the proof of Claim 39. \Box

Note that the final hybrid is identical to hybrid $Hyb_{2,k,2}$ and proof is done.

Claim 44. $Hyb_{2,k,2}$ is indistinguishable from $Hyb_{2,k,1}$.

Proof. Similar to the proof of Claim 35, we just need to prove that the extraction continues to hold. As we have already established the invariant, proving that the probability that $zap_{2,a}$ verifies only changes negligibly suffices to prove that the probability of well-formedness of the transcript only changes negligibly.

The proof of this is done similarly to Claim 35 using the sequence of sub-hybrids as in the proof of Claim 38. Note that, we only need 3 lookahead threads to extract the message in Ncom, the trapdoor, and the proof $zap_{2,a}$ in the reusable OT. Since we use a polynomial of degree-4 in our commitment schemes, it is indistinguishable to make those switches.

Claim 45. Assuming the privacy of the reusable MPC protocol, the invariant holds in $Hyb_{2,k,3}$.

Proof. The proof is similar to the proof of Claim 36. Note that every instance of the MPC protocol in each reuse session is independent of the others. Hence, we only rely on the external challenger to send us one message, i.e., the MPC message that we are switching. And we can simulate all other messages in the hybrid ourselves. \Box

Claim 46. Assuming the privacy of the reusable MPC protocol, $Hyb_{2,k,3}$ is indistinguishable from $Hyb_{2,k,2}$.

Proof. The proof is similar to the proof of Claim 37 and Claim 45.

Claim 47. Assuming the witness indistinguishability of the ZAP, the invariant holds in $Hyb_{2,k,4}$.

Proof. This proof is identical to the proof of Claim 36.

Claim 48. Assuming the witness indistinguishability of the ZAP, $Hyb_{2,k,4}$ is indistinguishable from $Hyb_{2,k,3}$.

Proof. This proof is identical to the proof of Claim 37.

Claim 49. Assuming the non-malleable with respect to extraction of Ncom, the invariant holds in $Hyb_{2,k,5}$.

Proof. This proof is identical to the proof of Claim 34.

Claim 50. Assuming the hiding property of the non-malleable commitment scheme, $Hyb_{2,k,5}$ is indistinguishable from $Hyb_{2,k,4}$.

Proof. This proof is identical to the proof of Claim 35.

Claim 51. Assuming the non-malleable with respect to extraction property of Ncom, the invariant holds in Hyb₃.

Proof. The proof is identical to the case when we switch the non-malleable commitment in the lookahead thread (Claim 34).

Claim 52. Assuming the hiding property of the Ncom, Hyb₃ is indistinguishable from Hyb₂.

Proof. If these two hybrids are not indistinguishable, then we shall break the hiding property of the non-malleable commitment.

Consider the adversary \mathcal{A}_{Ncom} that interacts with the external challenger. It simulates the hybrids exactly as described except for the non-malleable commitment protocol, which is exposed to the external challenger.

It runs 5 lookahead threads, which extract the input, trapdoor, and zap proof with non-negligible probability. Note that in the lookahead thread, the simulator is committing to a random string in the non-malleable commitment. We can ask the external challenger to provide those messages as the hiding property of the non-malleable commitment holds in the reusable setting.

Finally, it sends the external challenger the extracted trapdoor and a random string and receives back a commitment. This commitment is sent as the third round message of the non-malleable commitment for all future reuse sessions in the main thread. Depending on which messages are committed, we either simulate Hyb_3 or Hyb_2 . Hence, if the two hybrids are distinguishable, we break the hiding property of the non-malleable commitment.

Claim 53. Assuming the witness indistinguishable property of the ZAP, the invariant holds in Hyb_4 .

Proof. This proof is identical to the proof of Claim 36.

Claim 54. Assuming the witness indistinguishable property of the ZAP, Hyb_4 is indistinguishable from Hyb_3 .

Proof. If these two hybrids are distinguishable, we shall break the witness indistinguishable property of ZAP. The adversary \mathcal{A}_{zap} interacts with the external challenger. It simulates the hybrid exactly as described except for the zap protocol that we are switching. It runs 5 lookahead threads to do the extraction. Note that in those lookahead threads, we do not rely on the external challenger to generate any messages. Finally, it sends the challenger the statement and the honest witness, and the trapdoor witness. And the challenger replies with proof.

Depending on which witness the challenger uses, we either simulate Hyb_4 or Hyb_3 . Together with the efficient distinguisher for the two hybrids, we break the witness indistinguishability of the ZAP.

Claim 55. Assuming the witness indistinguishable property of the ZAP, the invariant holds in Hyb_5 .

Proof. Note that although we are modifying the fourth round of the main thread, it still might affect the invariant since there are future reuse sessions. However, this claim can be identically proven as in the proof of Claim 36.

Claim 56. Assuming the witness indistinguishable property of the ZAP, Hyb_5 is indistinguishable from Hyb_4 .

Proof. This proof is identical to the proof of Claim 54.

Claim 57. Assuming the security of rREcom, Hyb_6 is indistinguishable from Hyb_5 .

Proof. There is a potential circularity if we want to directly switch the input from honest to 0. In particular, we need at least 5 lookahead threads to extract the adversary's input. Note that we rely on the external challenger to generate the rREcom messages in the lookahead threads. However, given these many rewindings, switching the input from honest to 0 could be distinguishable. To avoid this circularity issue, we shall first modify the lookahead threads and instruct them to send junk messages in the last round of the rREcom. We shall argue that this does not harm extraction. Afterward, we can now switch the input from honest to 0 as we no longer rely on the external challenger to sample those third-round rREcom messages in the lookahead threads.

• In this hybrid, in the lookahead threads, the simulator sends $\{k \oplus q_i(0), q_i(z_i)\}_i$, SKE.Enc(k, 0) in the last round of the rREcom protocol for freshly sampled random polynomials q_1, \ldots, q_N .

Claim 58. The invariant holds in this hybrid.

Proof. We shall break the hiding property of Com if the invariant does not hold. The adversary \mathcal{A}_{Com} sends two polynomials p_i and q_i to the sender and the sender sends back a commitment of either p_i or q_i . \mathcal{A}_{Com} proceeds to sample the hybrid except for the commitment of the *i*-th polynomial. It samples the third round message of the rREcom as $\{k \oplus q_i(0), q_i(z_i)\}_i$, SKE.Enc(k, 0).

If the commitment that the challenger sends commits to p_i , we simulate this hybrid. If the commitment that the challenger sends commits to q_i , we simulate the previous hybrid.

 \mathcal{A}_{Com} proceeds to generate 2 lookahead threads to extract the message committed in Ncom using Ext_{NMCom} . If it is a valid trapdoor, it guesses 1. Otherwise, it guesses 0.

Since the invariant holds in the previous hybrid, if it does not hold in this hybrid, we break the hiding property of Com. $\hfill \Box$

Claim 59. This hybrid is indistinguishable from the previous one.

Proof. This proof is similar to the proof of Claim 59. The only difference is that we do not need to extract the message in the non-malleable commitment scheme but can directly use the distinguisher of the hybrids. \Box

• In this hybrid, we shall commit to 0 in the main thread.

Claim 60. The invariant holds in this hybrid.

Proof. If the invariant does not hold, we shall break the reusable security of rREcom.

The adversary \mathcal{A}_{rREcom} interacts with an external challenger. It simulates the hybrid as described except for the rREcom instance that we switch. It forwards all of its inputs to the challenger and the challenger is supposed to always commit to the honest input or 0 in all reuse sessions. Note that we do not rely on the external challenger to generate the third-round message of the rREcom protocol for us as they have already been switched to junk messages.

Next, \mathcal{A}_{rREcom} rewinds to sample one more thread to extract the message committed in Ncom. We will rely on the external challenger to sample those rREcom messages in this rewinding step. However, the polynomial is degree-4 and, hence, is secure against 1-rewinding.

Now, if the extracted message is a valid trapdoor, \mathcal{A}_{rREcom} outputs 1; otherwise, it outputs 0. If the invariant does not hold in this hybrid, \mathcal{A}_{rREcom} breaks the hiding property of rREcom. This completes the proof.

Claim 61. This hybrid is indistinguishable from the previous one.

Proof. The proof is similar to the proof of Claim 60. The only difference is that we do not need to extract the message in the non-malleable commitment scheme but can directly use the distinguisher of the hybrids. \Box

• In this hybrid, we switch the last round of the rREcom protocol in the lookahead threads back to honestly generated.

Claim 62. The invariant holds in this hybrid.

<i>Proof.</i> This proof is identical to the proof of Claim 58.	

Claim 63. This hybrid is indistinguishable from the previous one.

Proof. This proof is identical to the proof of Claim 59. \Box

Note that the last hybrid is exactly Hyb_6 and the proof is done.

Claim 64. Assuming the privacy of the reusable MPC protocol, the invariant holds in Hyb₇.

Proof. This proof is identical to the proof of Claim 45. Note that since the invariant holds in the previous hybrid, the adversary must be behaving in a semi-malicious manner if the proof verifies. \Box

Claim 65. Assuming the privacy of the reusable MPC protocol, Hyb_7 is indistinguishable from Hyb_6 . Note that since the invariant holds in the previous hybrid, the adversary must be behaving in a semi-malicious manner if the proof verifies.

Proof. This proof is similar to the proof of Claim 54.

Claim 66. Assuming the security of the garbled circuit, the invariant holds in Hyb_8 .

Proof. This is done by two hybrids.

• In this hybrid, we shall first change the four-round OT message. We use the extracted receiver's choice bits and prepare the sender's message using only the label corresponding to the receiver's choice bits. That is, in the step $rot_4^{i \to j} \leftarrow rOT_4(\{lab_{w,b}\}_{w \in P_j,b}, r_{rot}^{i \to j})$, only one label is provided for every input wire and the sender is using this label as both of its input messages.

Claim 67. The invariant holds in this hybrid.

Proof. If the invariant does not hold in this hybrid, we shall break the sender's security of the reusable OT protocol.

The adversary \mathcal{A}_{rOT} interacts with an external challenger. It simulates the hybrids as described except for the reusable OT protocol, which is exposed to the external challenger. It generates 5 lookahead threads to do the extraction. Note that we do not rely on the external challenger to generate the second sender's message in those lookahead threads as those messages are purely random strings.

Now it provides the challenger with both of the labels and the challenger is supposed to either use only one of them or both of them. Depending on which case the challenger selects, we either simulate this hybrid or the previous one.

Next, \mathcal{A}_{rOT} rewinds the adversary to extract the message in the Ncom using Ext_{NMCom}. (Again, we do not need the external challenger to generate the second-round rOT message as they are random strings) If the extracted value is a valid trapdoor, it guesses 1; otherwise, it guesses 0. If the invariant does not hold in this hybrid, this breaks the hiding property of the reusable OT protocol.

• In this hybrid, the simulator invokes the simulator of the garbled circuit with the output corresponding to either \perp or $\mathsf{rmpc}_{2,i}$ depending on implicit abort or no abort.

Claim 68. The invariant holds in this hybrid.

Proof. The proof is similar to the proof of Claim 67. Note that the garbled circuit is only used in the fourth round. Hence, the adversary has no issue generating the lookahead threads. \Box

Claim 69. Assuming the security of the garbled circuit, Hyb_8 is indistinguishable from Hyb_7 .

Proof. This proof is similar to the proof of Claim 66. The only difference is that we do not need to extract the committed message inside Ncom but can directly invoke the distinguisher for the hybrids to break the corresponding securities. \Box

Claim 70. The invariant holds in Hyb_{IDEAL}.

Proof. Since the main thread is identical to the previous one, the invariant still holds. \Box

Claim 71. Hyb_{IDEAL} and Hyb₈ is indistinguishable except with probability $\leq \frac{\mu}{4} + \operatorname{negl}(\lambda)$.

Proof. Suppose the probability that the adversary does not abort in the check abort step is $< \mu/4$, then the claim is correct as there is no difference when the adversary aborts.

Suppose the probability that the adversary does not abort is $\ge \mu/4$, then out of $5n\lambda/\mu$ threads, there will exist 5 GOOD threads with some party i^* with overwhelming probability. Additionally, the simulator will run in the expected polynomial time to get 12λ GOOD threads.

The only difference is that the simulator tries to re-sample the main thread for at most $\min(2^{\lambda}, \lambda^2/\varepsilon')$ times instead of λ/μ times. By the same argument as in [CCG⁺20], it can be shown that the two hybrids are indistinguishable in this case.

References

- [ACJ17] Prabhanjan Ananth, Arka Rai Choudhuri, and Abhishek Jain. A new approach to round-optimal secure multiparty computation. In Jonathan Katz and Hovav Shacham, editors, CRYPTO 2017, Part I, volume 10401 of LNCS, pages 468–499. Springer, Heidelberg, August 2017. 3
- [AJJM20] Prabhanjan Ananth, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Multi-key fullyhomomorphic encryption in the plain model. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 28–57. Springer, Heidelberg, November 2020. 3, 4, 9, 34
- [AJJM21] Prabhanjan Ananth, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Unbounded multiparty computation from learning with errors. In Anne Canteaut and François-Xavier Standaert, editors, EUROCRYPT 2021, Part II, volume 12697 of LNCS, pages 754–781. Springer, Heidelberg, October 2021. 3, 4, 9, 34
- [BGJ⁺18] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Amit Sahai. Promise zero knowledge and its applications to round optimal MPC. In Hovav Shacham and Alexandra Boldyreva, editors, CRYPTO 2018, Part II, volume 10992 of LNCS, pages 459–487. Springer, Heidelberg, August 2018. 3, 15, 40
- [BGMM20] James Bartusek, Sanjam Garg, Daniel Masny, and Pratyay Mukherjee. Reusable two-round MPC from DDH. In Rafael Pass and Krzysztof Pietrzak, editors, TCC 2020, Part II, volume 12551 of LNCS, pages 320–348. Springer, Heidelberg, November 2020. 3, 4, 9, 34
- [BGSZ22] James Bartusek, Sanjam Garg, Akshayaram Srinivasan, and Yinuo Zhang. Reusable two-round MPC from LPN. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, Public-Key Cryptography - PKC 2022 - 25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8-11, 2022, Proceedings, Part I, volume 13177 of Lecture Notes in Computer Science, pages 165–193. Springer, 2022. 3, 4, 34

- [BHP17] Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. In Yael Kalai and Leonid Reyzin, editors, TCC 2017, Part I, volume 10677 of LNCS, pages 645–677. Springer, Heidelberg, November 2017. 3
- [BJKL21] Fabrice Benhamouda, Aayush Jain, Ilan Komargodski, and Huijia Lin. Multiparty reusable non-interactive secure computation from LWE. In Anne Canteaut and François-Xavier Standaert, editors, EUROCRYPT 2021, Part II, volume 12697 of LNCS, pages 724–753. Springer, Heidelberg, October 2021. 3, 4, 9, 34
- [BJY97] Mihir Bellare, Markus Jakobsson, and Moti Yung. Round-optimal zero-knowledge arguments based on any one-way function. In Walter Fumy, editor, EUROCRYPT'97, volume 1233 of LNCS, pages 280–305. Springer, Heidelberg, May 1997. 5
- [BL18] Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, EUROCRYPT 2018, Part II, volume 10821 of LNCS, pages 500–532. Springer, Heidelberg, April / May 2018. 3
- [BL20] Fabrice Benhamouda and Huijia Lin. Mr NISC: Multiparty reusable non-interactive secure computation. In Rafael Pass and Krzysztof Pietrzak, editors, TCC 2020, Part II, volume 12551 of LNCS, pages 349–378. Springer, Heidelberg, November 2020. 3, 4, 9, 34
- [BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In 22nd ACM STOC, pages 503–513. ACM Press, May 1990. 3
- [CCG⁺20] Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Round optimal secure multiparty computation from minimal assumptions. In Rafael Pass and Krzysztof Pietrzak, editors, TCC 2020, Part II, volume 12551 of LNCS, pages 291–319. Springer, Heidelberg, November 2020. 3, 9, 15, 16, 32, 33, 34, 35, 37, 38, 40, 42, 43, 50
- [CCG⁺21] Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Oblivious transfer from trapdoor permutations in minimal rounds. In Kobbi Nissim and Brent Waters, editors, Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part II, volume 13043 of Lecture Notes in Computer Science, pages 518–549. Springer, 2021. 3
- [DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *41st FOCS*, pages 283–293. IEEE Computer Society Press, November 2000. 6, 15
- [FJK21] Rex Fernando, Aayush Jain, and Ilan Komargodski. Maliciously-secure MrNISC in the plain model. Cryptology ePrint Archive, Report 2021/1319, 2021. https://eprint.iacr.org/2021/1319. 3
- [FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In 31st FOCS, pages 308–317. IEEE Computer Society Press, October 1990. 5
- [FMV19] Daniele Friolo, Daniel Masny, and Daniele Venturi. A black-box construction of fully-simulatable, round-optimal oblivious transfer from strongly uniform key agreement. In Dennis Hofheinz and Alon Rosen, editors, TCC 2019, Part I, volume 11891 of LNCS, pages 111–130. Springer, Heidelberg, December 2019. 3, 7
- [GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In 14th ACM STOC, pages 365–377. ACM Press, May 1982. 23
- [GMPP16] Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 448–476. Springer, Heidelberg, May 2016. 3, 4

- [GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, 19th ACM STOC, pages 218–229. ACM Press, May 1987. 3
- [GPR16] Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. In Daniel Wichs and Yishay Mansour, editors, 48th ACM STOC, pages 1128–1141. ACM Press, June 2016. 34
- [GS18] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, EUROCRYPT 2018, Part II, volume 10821 of LNCS, pages 468–499. Springer, Heidelberg, April / May 2018. 3
- [HHPV18] Shai Halevi, Carmit Hazay, Antigoni Polychroniadou, and Muthuramakrishnan Venkitasubramaniam. Round-optimal secure multi-party computation. In Hovav Shacham and Alexandra Boldyreva, editors, CRYPTO 2018, Part II, volume 10992 of LNCS, pages 488–520. Springer, Heidelberg, August 2018. 3
- [HLP11] Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In Phillip Rogaway, editor, CRYPTO 2011, volume 6841 of LNCS, pages 132–150. Springer, Heidelberg, August 2011. 3
- [IKO⁺11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In Kenneth G. Paterson, editor, EUROCRYPT 2011, volume 6632 of LNCS, pages 406–425. Springer, Heidelberg, May 2011. 3
- [IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, CRYPTO 2008, volume 5157 of LNCS, pages 572–591. Springer, Heidelberg, August 2008. 3
- [KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew Franklin, editor, CRYPTO 2004, volume 3152 of LNCS, pages 335–354. Springer, Heidelberg, August 2004. 3, 4, 7, 28
- [Nao91] Moni Naor. Bit commitment using pseudorandomness. J. Cryptol., 4(2):151–158, 1991. 8, 28
- [NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, 12th SODA, pages 448–457. ACM-SIAM, January 2001. 22, 23
- [ORS15] Rafail Ostrovsky, Silas Richelson, and Alessandra Scafuro. Round-optimal black-box two-party computation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, CRYPTO 2015, Part II, volume 9216 of LNCS, pages 339–358. Springer, Heidelberg, August 2015. 3, 7
- [PW09] Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In Omer Reingold, editor, TCC 2009, volume 5444 of LNCS, pages 403–418. Springer, Heidelberg, March 2009. 14
- [Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In 22nd ACM STOC, pages 387–394. ACM Press, May 1990. 15
- [Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In 27th FOCS, pages 162–167. IEEE Computer Society Press, October 1986. 3, 5