Promise Zero Knowledge and its Applications to Round Optimal MPC

Saikrishna Badrinarayanan
UCLA
saikrishna@cs.ucla.edu

Vipul Goyal
CMU
goyal@cs.cmu.edu

Abhishek Jain
JHU
abhishek@cs.jhu.edu

Yael Tauman Kalai
Microsoft Research, MIT
yael@microsoft.com

Dakshita Khurana
UCLA
dakshita@cs.ucla.edu

Amit Sahai
UCLA
sahai@cs.ucla.edu

Abstract

We devise a new *partitioned simulation* technique for MPC where the simulator uses different strategies for simulating the view of aborting adversaries and non-aborting adversaries. The protagonist of this technique is a new notion of *promise zero knowledge* (ZK) where the ZK property only holds against non-aborting verifiers. We show that unlike standard ZK, promise ZK can be realized in three rounds in the simultaneous-message model under standard assumptions.

We demonstrate the following applications of our new technique:

- We construct the first round-optimal (i.e., four round) MPC protocol for general functions based on polynomially hard LWE and DDH. Previously, such protocols required sub-exponential-time hardness assumptions.
- We further show how to overcome the four-round barrier for MPC by constructing a three-round protocol for "list coin-tossing" – a slight relaxation of coin-tossing that suffices for most conceivable applications – based on polynomially hard injective one-way functions. This result generalizes to randomized input-less functionalities also assuming polynomially hard LWE.

Previously, four round MPC protocols required sub-exponential hardness assumptions and no multi-party three-round protocols with polynomial simulation were known for any relaxed security notions against malicious adversaries.

In order to base security on polynomial-time standard assumptions, we also develop a new *leveled rewinding security* technique that can be viewed as a polynomial-time alternative to leveled complexity leveraging for achieving "non-malleability" across different primitives. This technique may be of independent interest.

Contents

1 Introduction

Provably secure protocols lie at the heart of the theory of cryptography. How can we design protocols, not only so that we cannot devise attacks against them, but so that we can *prove* that no such attacks exist (under well-studied complexity assumptions)? The goal of achieving a proof of security has presented many challenges and apparent trade-offs in secure protocol design. This is especially true with regards to the goal of minimizing rounds of interaction, which has been a long-standing driver of innovation in theoretical cryptography. We begin by focusing on one such challenge and apparent trade-off in the context of *zero-knowledge* (ZK) protocols [GMR89], one of the most fascinating and broadly applicable notions in cryptography.

Recall that in a ZK protocol, a prover should convince a verifier that some statement is true, without revealing to the verifier anything beyond the validity of the statement being proven. It is known that achieving zero knowledge (with black-box simulation[1]) is impossible with three or fewer rounds of simultaneous message exchange [GK96, GMPP16]. A curious fact emerges, however, when we take a closer look at the proof of this impossibility result. It turns out that three-round ZK is impossible when considering verifiers that essentially behave completely honestly, but that sometimes probabilistically refuse to finish the protocol. We observe that this is bizarre: ZK protocols are supposed to prevent the verifier from learning information from the prover; how can behaving honestly but aborting the protocol early possibly help the verifier learn additional information? Indeed, one might think that we can prove that such behavior cannot possibly help the verifier learn additional information. Counter-intuitively, however, it turns out that such early aborts are critical to the impossibility proofs of [GK96, GMPP16]. This observation is the starting point for our work; now that we have identified a key (but counter-intuitive) reason behind the impossibility results, we want to leverage this understanding to bypass the impossibility result in a new and useful way.

Promise Zero Knowledge. Our main idea is to circumvent the impossibility results of [GK96, GMPP16] by only considering adversarial verifiers that promise not to abort the protocol early with noticeable probability. However, we do not limit ourselves only to adversarial verifiers that behave honestly; we will consider adversarial verifiers that may deviate from the prescribed protocol arbitrarily, as long as this deviation does not cause the protocol to abort. A *promise zero-knowledge* protocol is one that satisfies the correctness and soundness guarantees of ordinary zero-knowledge protocols, but only satisfies the zero knowledge guarantee against adversarial verifiers that "promise" not to abort with noticeable probability. The centerpiece of our work is a construction of a three-round promise zero-knowledge protocol, in the simultaneous message model, for proving statements where the statement need not be decided until the last (third) round, but where such statements should come from a distribution such that both a statement and a witness for that statement can be sampled in the last round. Our construction requires only the existence of injective one-way functions. Interestingly, in our construction, we rely upon information learned from the verifier in the third round, to simulate its view in the third round!

Partitioned Simulation, and Applications to MPC. But why should we care about promise ZK? Actual adversaries will not make any promise regarding what specific types of adversarial behavior they will or will not engage in. However, recall our initial insight – early aborting by an

[1] In this work, we focus on black-box simulation. However, no solutions for three-round ZK from standard assumptions with non-black-box simulation [Bar01] are presently known either. [BKP17] showed how to construct 3 round ZK using non-black-box simulation from the non-standard assumption that keyless multi-collision resistant hash functions exist.

adversary should, generally speaking, only hurt the adversary, not help it. We know due to the impossibility results of [GK96, GMPP16], that we cannot leverage this insight to achieve three-round standard ZK (with black-box simulation). However, ZK is a ubiquitous technical tool used to construct secure protocols. Our goal instead, then, is to use our insight to replace ZK with promise ZK for the construction of other secure protocols. Specifically, we consider the most general goal of secure protocol design: secure multi-party computation (MPC), as we discuss further below.

To do so, we devise a novel *partitioned simulation* strategy for leveraging promise ZK. At a high-level, we split the simulation into two disjoint cases, *depending upon whether or not the adversary is an aborting adversary* (i.e., one who aborts with high probability). In one case, we will exploit promise ZK. In the other, we exploit the intuition that early aborting should only harm the adversary, to devise alternate simulation strategies that bypass the need for ZK altogether, and instead essentially rely on a weaker notion called strong witness indistinguishability, that was recently constructed in three rounds (in the "delayed-input" setting) in [JKKR17].

Secure Multi-Party Computation. The notion of secure multiparty computation (MPC) [Yao82, GMW87] is a unifying framework for general secure protocols. MPC allows mutually distrusting parties to jointly evaluate any efficiently computable function on their private inputs in such a manner that each party does not learn anything beyond the output of the function.

The round complexity of MPC has been extensively studied over the last three decades in a long sequence of works [GMW87, BMR90, KOS03, KO04, Pas04, PW10, Wee10, Goy11, GMPP16, ACJ17, BHP17, COSV17a, COSV17b]. In this work, we study the problem of *round-optimal* MPC against malicious adversaries who may corrupt an arbitrary subset of parties, in the plain model without any trust assumptions. The state-of-the-art results on round-optimal MPC for general functions are due to Ananth et al. [ACJ17] and Brakerski et al. [BHP17], both of which rely on sub-exponential-time hardness assumptions. (See Section 1.2 for a more elaborate discussion on related works.) Our goal, instead is to base security on standard, *polynomial-time* assumptions.

We now highlight the main challenge in basing security on polynomial-time assumptions. In the setting of four round protocols in the simultaneous-message model, a rushing adversary may always choose to abort after receiving the honest party messages in the last round. At this point, the adversary has already received enough information to obtain the purported output of the function being computed. This suggests that we must enforce "honest behavior" on the parties within the first three rounds in order to achieve security against malicious adversaries. As discussed above, three-round zero knowledge is impossible, and this is precisely why we look to our new notion of promise ZK and partitioned simulation to resolve this challenge.

However, this challenge is exacerbated in the setting of MPC as we must not only enforce honest behavior but also ensure non-malleability *across different cryptographic primitives* that are being executed in parallel within the first three rounds. We show how to combine our notions of promise ZK with new simulation ideas to overcome these challenges, relying only on polynomial-time assumptions.

Coin Tossing. Coin-tossing allows two or more participants to agree on a single, unbiased coin. Fair multiparty coin-tossing is known to be impossible in the dishonest majority setting [Cle86]. Therefore while current notions of *secure coin-tossing* require that the protocol have a (pseudo)-random outcome, the adversary is additionally allowed to abort depending on the outcome of the toss.

The definition of secure coin-tossing roughly requires the existence of a simulator that successfully forces an externally sampled random coin, and produces a distribution over adversary's views

that is indistinguishable from a real execution. To account for the adversary aborting or misbehaving based on the outcome, the simulator is allowed to either force an external coin, or force an abort: as long as the simulated distribution remains indistinguishable from the real one.

In the case of an adversary that always aborts before the end of the protocol, the prescribed output of any secure coin-tossing protocol is also abort: therefore, the simulator never needs to force *any external coin* against such an adversary! Simulating the view of such adversaries that always abort is thus completely trivial.

This leaves open the setting of non-aborting adversaries, which is exactly the setting that promise ZK was designed for. Using promise ZK, we design a three-round coin-tossing protocol which crucially relies on the promise ZK simulator to ensure security against non-aborting adversaries. Previously, multiparty coin-tossing was known to require at least four rounds w.r.t. black-box simulation [GMPP16, KO04].

For technical reasons, we achieve a slightly weaker notion of coin-tossing than prior literature, which we call "list coin-tossing". As we discuss later, this notion nevertheless suffices for nearly all important applications of coin-tossing. Therefore, promise ZK gives us a way to break down the four-round barrier for secure coin-tossing [GMPP16, KO04].

1.1 Our Results

We introduce the notion of promise ZK proof systems and devise a new partitioned simulation strategy for round-efficient MPC protocols. Our first result is a three-round distributional promise ZK argument system based on injective one-way functions.

Theorem 1 (Informal)**.** *Assuming injective one-way functions, there exists a three round distributional promise ZK argument system in the simultaneous-message model.*

Round-Optimal MPC. We present two applications of partitioned simulation to round-optimal MPC. We first devise a general compiler from any three round semi-malicious MPC protocol – where the first round is immune to malicious behavior – to a four round malicious secure MPC protocol. Our compiler can be instantiated with standard assumptions such as DDH or Quadratic Residuosity or N^{th}-Residuosity. The resulting protocol is optimal in the number of rounds w.r.t. black-box simulation [GMPP16]. A three round semi-malicious protocol with the aforementioned property is known based on LWE [BHP17].

Theorem 2 (Informal)**.** *Assuming LWE and DDH/QR/N^{th}-Residuosity, there exists a four round MPC protocol for general functions with black-box simulation.*

List Coin-Tossing. We also study the feasibility of multiparty coin-tossing in only three rounds. While three round coin-tossing is known to be impossible [GMPP16], somewhat surprisingly, we show that a slightly relaxed variant that we refer to as *list coin-tossing* is, in fact, possible in only three rounds.

Very briefly, in list coin-tossing, the simulator is allowed to receive polynomially many random string samples from the ideal functionality (where the exact polynomial may depend upon the adversary), and it may choose any one of them as its output. It is not difficult to see that this notion already suffices for most conceivable applications of coin-tossing, such as implementing a common random string setup. For example, consider the setting where we want to generate a CRS in the setup algorithm of a non-interactive zero knowledge (NIZK) argument system. Now, in the ideal world, instead of running a simulator which "forces" one particular random string given by the ideal functionality, we can substitute it with the simulator of a list coin tossing protocol that

receives polynomially many random strings from the ideal functionality and "forces" one of them as the CRS. This would still suffice for the NIZK argument system. We achieve the following result:

Theorem 3 (Informal)**.** *Assuming injective one-way functions, there exists a three round multiparty list coin-tossing protocol with black-box simulation.*

With the additional assumption of LWE, we can also generalize the above result to randomized inputless functionalities where security is defined analogously to list coin-tossing.

Finally, we note that by applying the transformation[2] of [GMPP16] on the protocol from Theorem 3 for the two-party case, we can obtain a four round two-party list coin-tossing protocol in the *unidirectional-message* model. This result overcomes the barrier of five rounds for standard two-party coin-tossing established by [KO04].

Corollary 4 (Informal)**.** *Assuming injective one-way functions, there exists a four round two-party list coin-tossing protocol in the unidirectional-message model with black-box simulation.*

Leveled Rewinding Security. While promise ZK addresses the issue of proving honest behavior within three rounds, it does not address non-malleability issues that typically plague security proofs of constant-round protocols in the simultaneous-message model. In particular, when multiple primitives are being executed in parallel, we need to ensure that they are non-malleable w.r.t. each other. For example, we may require that a primitive A remains "secure" while the simulator (or a reduction) is (say) trying to extract adversary's input from primitive B via rewinding.

In the works of [ACJ17, BHP17], such issues are addressed by using complexity leveraging. In particular, they rely upon multiple levels of complexity leveraging to establish non-malleability relationships across primitives, e.g., by setting the security parameters such that primitive X is more secure than primitive Y that is more secure than primitive Z, and so on. Such a use a complexity leveraging is, in fact, quite common in the setting of limited rounds (see, e.g., [COSV16]).

We develop a new *leveled rewinding security* technique to avoid the use of complexity leveraging and base security on polynomial-time assumptions. Roughly, in our constructions, primitives have various levels of "bounded rewinding" security that are carefully crafted so that they enable non-malleability relationships across primitives, while still enabling rewinding-based simulation and reductions. E.g., a primitive X may be insecure w.h.p. against 1 rewind, however, another primitive Y may be secure against 1 rewind but insecure against 2 rewinds. Yet another primitive Z may be secure against 2 rewinds but insecure against 3 rewinds, and so on. We anticipate that this technique will find applications elsewhere in cryptography.

1.2 Related Work

Concurrent Work. In a concurrent and independent work, Halevi et al. [HHPV17] construct a four round MPC protocol against malicious adversaries in the plain model based on LWE/DDH/QR/N^{th}-Residuosity. They do not consider the problems of promise ZK and list coin-tossing. We will add a more detailed comparison between the works and the techniques in a revised version once we have had a chance to review their paper.

Prior Work. The study of constant-round protocols for MPC was initiated by Beaver et al. [BMR90]. They constructed constant-round MPC protocols in the presence of honest majority.

[2]The work of Garg et al. [GMPP16] establishes an impossibility result for three round multiparty coin-tossing by transforming any three round two-party coin-tossing protocol in the simultaneous-message model into a four round two-party coin-tossing protocol in the unidirectional-message model, and then invoking the impossibility of [KO04].

Subsequently, a long sequence of works constructed constant-round MPC protocols against dishonest majority based on a variety of assumptions and techniques (see, e.g., [KOS03, Pas04, PW10, Wee10, Goy11]).

Garg et al. [GMPP16] initiated the study of the exact round complexity of MPC. They constructed five (resp., six) round MPC using indistinguishability obfuscation (resp., LWE) and three-round robust non-malleable commitments. Recently, in concurrent works, Ananth et al. [ACJ17] and Brakerski et al. [BHP17] constructed four round MPC protocol based on sub-exponential-time hardness assumptions. While [BHP17] require sub-exponential LWE and adaptive commitments [PPV08], the work of [ACJ17] additionally relied on the recent construction of two-round non-malleable commitments from [KS17] to obtain a construction from sub-exponential DDH.

Very recently, [COSV17a] constructed four-round multiparty coin-tossing from polynomial-time assumptions. The same authors also construct four-round two-party computation in the simultaneous-message model from polynomial-time assumptions [COSV17b]. However, their results do not extend to general multiparty functionalities, which is the focus of our work.

In the setting of super-polynomial-time simulation, [BGI+17, JKKR17] construct two round two-party computation from sub-exponential DDH. In the multi-party setting, [KS17] construct coin-flipping and [BGJ+17] extend this result to input-less randomized functionalities. Garg et al. [GKP17] construct five round concurrent two-party computation from quasi-poly hard assumptions and [BGJ+17] construct three round concurrent MPC from sub-exponential LWE.

All of the above results are in the plain model where no trusted setup assumptions are available. Asharov et. al. [AJL+12] constructed three round MPC protocols in the CRS model. Subsequently, two-round MPC protocols in the CRS model were constructed by Garg et al. [GGHR14] using indistinguishability obfuscation, and by Mukherjee and Wichs [MW16] using LWE assumption. Very recently, Garg and Srinivasan [GS17] constructed a two-round MPC protocol based on standard assumptions on bilinear maps.

2 Technical Overview

In this section, we provide an overview of the main ideas underlying our results.

2.1 Promise Zero Knowledge

Recall that the notion of promise ZK is defined in the simultaneous-message model, where in every round, both the prover and the verifier send a message simultaneously.[3] Crucially, the ZK property is only defined w.r.t. a set of admissible verifiers that promise to send a "valid" non-aborting message in the last round with some noticeable probability.

We construct a three round distributional promise ZK protocol with black-box simulation based on injective one-way functions. We work in the delayed-input setting where the statement being proven is revealed to the (adversarial) verifier only in the last round.[4] Further, we work in the distributional setting, where statements being proven are from an efficiently sampleable public distribution, i.e., it is possible to efficiently sample a statement together with a witness.

For simplicity of presentation, here we describe our construction using an additional assumption of two-round WI proofs, a.k.a. Zaps [DN00]. In our actual construction of promise ZK, we replace

[3] An adversarial prover or verifier can be rushing, i.e., it may wait to receive a message from the honest party in any round before sending its own message in that round.

[4] In our actual construction, we consider a slightly more general setting where a statement x has two parts (x_1, x_2): the first part x_1 is revealed in the second round while the second part x_2 is revealed in the third round. This generalization is used in our applications of promise ZK, but we ignore it here for simplicity of presentation.

the Zaps with three round delayed-input WI proofs with some additional security guarantees that we construct based on injective one-way functions.[5]

Our construction of promise ZK roughly follows the FLS paradigm [FLS90] for ZK:

- First, the prover and the verifier engage in a three round "trapdoor generation phase" that determines a secret "trapdoor" that is known to the verifier but not the prover.
- Next, in a proof phase, the prover commits to 0 in a (three round) delayed-input extractable commitment and proves via a Zap that either the purported statement is true or that it committed to the trapdoor (instead of 0).

By appropriately parallelizing both of these phases, we obtain a three round protocol in the simultaneous-message model. Below, we discuss the challenges in proving soundness and promise ZK properties.

Proving Soundness. In order to argue soundness, a natural strategy is to rewind the cheating prover in the second and third round to extract the value it has committed in the extractable commitment. If this value is the trapdoor, then we can (hopefully) break the hiding property of the trapdoor generation phase to obtain a contradiction. Unfortunately, this strategy doesn't work as is since the trapdoor generation phase is parallelized with the extractable commitment. Thus, while extracting from the extractable commitment, we may inadvertently also break the security of the trapdoor generation phase! Indeed, this is the key problem that arises in the construction of non-malleable protocols.

Our main observation is that in order to prove soundness, it suffices to extract the trapdoor from the cheating prover with some noticeable probability (as opposed to overwhelming probability). Now, suppose that the extractable commitment scheme is such that it is possible to extract the committed value via k rewinds (for some small integer k) if the "main thread" of execution is non-aborting with noticeable probability. Then, we can still argue soundness if the trapdoor generation has a stronger hiding property, namely, security under k rewinds (but is insecure under more than k rewinds to enable simulation; see below).

We note that standard extractable commitment schemes such as [PRS02, Ros04] achieve the above extraction property for $k = 1$. This means that we only require the trapdoor generation phase to maintain hiding property under 1 rewinding. Such a scheme can be easily constructed from one-way functions.

Proving Promise ZK. In order to prove the promise ZK property, we construct a simulator that learns information from the verifier in the third round, in order to simulate its view in the third round! Roughly, our simulator first creates multiple "look-ahead" execution threads[6] with the adversarial verifier in order to extract the trapdoor from the trapdoor generation phase. Note that unlike typical ZK protocols where such a look-ahead thread only consists of partial protocol transcript, in our case, each look-ahead thread must contain a full protocol execution since the trapdoor generation phase completes in the third round.

Now, since the adversarial verifier may be rushing, the simulator must first provide its third round message (namely, the second message of Zap) on each look-ahead thread in order to learn the verifier's third round message. Since the simulator does not have a trapdoor yet, the only possibility for the simulator to prepare a valid third round message is by behaving honestly. However, the

[5] In particular, replacing Zaps with delayed-input WI proofs relies on our *leveled rewinding security* technique that we describe in Section 2.2. We do not discuss it here to avoid repetition.

[6] Throughout, whenever the simulator rewinds, we call each rewound execution a look-ahead thread. The messages that are eventually output by the simulator are denoted by the main thread.

simulator does not have a witness for the statement proven by the honest prover. Thus, it may seem that we have run into a circularity.

This is where the distributional aspect of our notion comes to the rescue. Specifically, on the look-ahead execution threads, the simulator simply samples a fresh statement together with a witness from the distribution and proves the validity of the statement like an honest prover. Once it has extracted the trapdoor, it uses its knowledge to cheat (only) on the main thread (but continues to behave honestly on each look-ahead thread). [7]

2.2 Four Round Secure Multiparty Computation

We now describe the main ideas underlying our compiler from any three round semi-malicious MPC protocol Π (where the first round is immune to malicious behavior) to a four round malicious-secure MPC protocol Σ. For simplicity of presentation, in the discussion below, we ignore the first round of Π, and simply treat it as a *two round* protocol.

Starting Ideas. Similar to [ACJ17], our starting idea is to follow the GMW paradigm [GMW87] for malicious security. This entails two main steps: (1) Enabling extraction of adversary's inputs, and (2) Forcing honest behavior on the adversary in each round of Π. A natural idea to implement the first step is to require each party to commit to its input and randomness via a three round extractable commitment protocol. To force honest behavior, we require each party to give a delayed-input ZK proof together with every message of Π to establish that it is "consistent" with the input and randomness committed in the extractable commitment.

In order to obtain a four-round protocol Σ, we need to parallelize all of these sub-protocols appropriately. This means that while the proof for the second message of Π can be given via a four round (delayed-input) regular ZK proof, we need a *three round* proof system to prove the well-formedness of the first message of Π. However, as discussed earlier, three-round ZK proofs are known to be impossible w.r.t. black-box simulation [GK96, GMPP16] and even with non-black box simulation, are not known from standard assumptions.

Promise ZK and Partitioned Simulation. While Ananth et al. [ACJ17] tackled this issue by using sub-exponential hardness, we address it via partitioned simulation to base security on polynomial-time assumptions. Specifically, we use different mechanisms for proving honest behavior depending upon whether or not the adversary is aborting in the third round. For now, let us assume that the adversary does not abort in the third round of Σ; later we briefly discuss the aborting adversary case.

For the non-aborting case, we rely upon a three-round (delayed-input) distributional promise ZK to prove well-formedness of the first message of Π. As we discuss below, however, integrating promise ZK in our construction involves overcoming several technical challenges due to specific properties of the promise ZK simulator (in particular, its requirement to behave honestly in look-ahead threads).[8]

We also remark that in our actual construction, to address non-malleability concerns [DDN91], the promise ZK and the standard ZK protocols that we use are suitably "hardened" using three-round non-malleable commitments [GPR16, Khu17] to achieve *simulation soundness* [Sah99] in

[7] The idea of using a witness to continue simulation is an old one[BS05]. Most recently, [JKKR17] used this idea in the context of arguments in the distributional setting.

[8] Our construction of four round MPC, in fact, uses promise ZK in a non-black-box manner for technical reasons. We ignore this point here as it is not important to the discussion.

order to ensure that the proofs given by the adversarial parties remain sound even when the proofs given by honest parties are simulated.

For simplicity of discussion, however, here we largely ignore this point, and instead focus on the technical ideas that are more unique to our construction.

How to do "Non-Malleable" Input Extraction? While the above serves as a good starting point, things start to unravel quickly when we attempt to prove security. Let us start with the issue of extraction of adversary's input and trapdoors (for simulation of ZK proofs). In the above protocol, in order to extract adversary's input and trapdoors, the simulator rewinds the second and third rounds. Note, however, that this process also rewinds the input commitments of the honest parties since they are executed in parallel. This poses the following fundamental challenge: we must somehow maintain privacy of honest party's inputs *even under rewinds*, while still extracting the inputs of the adversarial parties.

A plausible strategy to address this issue is to cheat in the rewound executions by sending random third round messages in the input commitment protocol on behalf of each honest party. This effectively nullifies the effect of rewinding on the honest party input commitments. However, in order to implement such a strategy, we need the ability to cheat in the ZK proofs since they are proving "well-formedness" of the input commitments.[9]

Unfortunately, such a strategy is not viable in our setting. As discussed in the previous subsection, in order to simulate the promise ZK on the main thread, the simulator must behave "honestly" on the rewound execution threads. This suggests that we cannot simply "sidestep" the issue of rewinding and instead must somehow make the honest party input commitments immune to rewinding. Yet, we must do this while still keeping the adversary input commitments extractable. Thus, it may seem that we have reached an impasse.

Leveled Rewinding Security to the Rescue. In order to break the symmetry between input commitments of honest and adversarial parties, we use the following sequence of observations:

- The security of the honest party input commitments is only invoked when we switch from a hybrid experiment (say) H_i to another experiment H_{i+1} inside our security proof. In order to argue indistinguishability of H_i and H_{i+1} by contradiction, it suffices to build an adversary that breaks the security of honest party input commitments with some noticeable probability (as opposed to overwhelming probability).
- This means that the reduction only needs to generate the view of the adversary in hybrids H_i and H_{i+1} with some noticeable probability. This, in turn, means that the reduction only needs to successfully extract the adversary's inputs and trapdoor (for generating its view) with noticeable probability.
- Now, recall that the trapdoor generation phase used in our promise ZK construction is secure against one rewind. However, if we rewind two times, then we can extract the trapdoor with noticeable probability.
- Now, suppose that we can construct an input commitment protocol that maintains hiding property even if it is rewound two times, but guarantees extraction with noticeable probability if it is rewound three times. Given such a commitment scheme, we resolve the above problem as follows: the reduction rewinds the adversary three times, which ensures that with noticeable probability, it can extract *both* the trapdoor and the inputs from the adversary. In the first

[9] Indeed, [ACJ17] implement such a strategy in their security proof by relying on sub-exponential hardness assumptions.

two rewound executions, the reduction generates the third round messages of the honest party input commitments honestly. At this point, the reduction already has the trapdoor. Now, in the third rewound execution, it generates random third messages in the honest party input commitments and uses the knowledge of the trapdoor to cheat in the proof.

The above strategy allows us to extract the adversary's inputs with noticeable probability while still maintaining privacy of honest party inputs. To complete this idea, we construct a new extractable commitment scheme from injective one-way functions that achieves the desired "bounded-rewinding" property.

Taking a step back, note that in order to implement the above strategy, we created two levels of rewinding security: while the trapdoor generation phase is secure against one rewind (but insecure against two rewinds), the input commitment protocol is secure against two rewinds (but insecure against three rewinds). We refer to this technique as *leveled rewinding security*, and this is precisely what allows us to avoid the use of leveled complexity leveraging used in [ACJ17].

A Double-Rewinding Strategy. Similar to [ACJ17], our final simulator also behaves honestly in the first three rounds using *random* inputs for the honest parties. The role of promise ZK is to help transition from honest behavior using real inputs to honest behavior using random inputs.

Unfortunately, it is not immediately clear how to implement the above idea in our setting. The main issue arises due to the properties of the simulator of promise ZK: in order to facilitate switching from using input (say) x_i to input 0 for an honest party i on the main thread, we create rewound execution threads where we behave honestly using input x_i (this is required by the simulator of promise ZK). This means that in our final simulation, we need to remove the use of input x_i from *every execution thread* and not just the main thread. However, for any such switch on any execution thread, we need some execution threads where we behave honestly. So once again, it seems that we have run into a circularity.

We resolve the above issue by using a *double-rewinding strategy*. Roughly, suppose we were using t rewound execution threads to facilitate switching from input x_i to input 0 on the main thread. Then, we double the number of rewound execution threads to $2t$. Now, we behave honestly on the first t threads as usual, but on each thread $t + j$, where $1 \leq j \leq t$, we switch from input x_i to 0 one-by-one. To facilitate each of these transitions, we utilize the first t threads. Once we have successfully made the switch on each thread $t + j$, we simply "kill" the first t threads, and now we are simply left with t rewound execution threads where we use input 0.

Other Issues. The above discussion ignores several important details. For one, so far we haven't addressed the case where the adversary aborts in the third round with overwhelming probability. Note that in this case, we cannot rely upon promise ZK since there is no hope for extraction from such an aborting adversary (which is necessary for simulating promise ZK).

Thus, we need a different mechanism in this case to allow us to transition from honest behavior in the first three rounds using real inputs to honest behavior using random inputs. On the positive side, in this case, we do not need any of our sub-protocols to be "rewinding secure" since there is no rewinding happening at all in this case. For this reason, we are able to rely upon the techniques from the recent work of Jain et al. [JKKR17] to address this specific case. We are able to adapt their ideas for constructing three round strong WI arguments to our setting without much modifications; however, as in their work, the proofs are quite delicate, and we refer the reader to the technical sections for more details.

Finally, we note that since our partitioned simulation technique crucially relies upon identifying whether an adversary is aborting or not, we have to take precaution during simulation to avoid

the possibility of the simulator running in exponential time. For this reason, we use ideas first developed in [GK96] and later used in many subsequent works, to ensure that the running time of our simulator is expected polynomial-time.

2.3 List Coin-Tossing

We now describe the main ideas underlying our construction of three round multiparty list coin-tossing. We start by describing the basic structure of our protocol:

- We start with a two-round semi-honest multiparty coin-tossing protocol based on injective one-way functions. Such a protocol can be constructed as follows: in the first round, each party i commits to a string r_i chosen uniformly at random, using a non-interactive commitment scheme. In the second round, each party reveals r_i without the decommitment information. The output is simply the XOR of all the r_i values.
- To achieve malicious security, we "compile" the above semi-honest protocol with a (delayed-input) distributional promise ZK protocol. Roughly speaking, in the third round, each party i now proves that the value r_i is the one it had committed earlier. By parallelizing the two sub-protocols appropriately, we obtain a three round protocol.

We first note that as in the case of our four round MPC protocol, here also we need to "harden" the promise ZK protocol with non-malleability properties. We do so by constructing a three-round simulation-extractable promise ZK based on injective one-way functions and then using it in the above compiler. Nevertheless, for simplicity of discussion, we do not dwell on this issue here, and refer the reader to the technical sections for further details.

We now describe the main ideas underlying our simulation technique. As in the case of four round MPC, we use partitioned simulation strategy to split the simulation into two cases, depending upon whether the adversary aborts or not in the third round.

Aborting Case. If the adversary aborts in the third round, then the simulator simply behaves honestly using a uniformly random string r_i on behalf of each honest party i. Unlike the four round MPC case, indistinguishability can be argued here in a straightforward manner since the simulated transcript is identically distributed as a real transcript. The main reason why such a strategy works is that since the parties do not have any input, there is no notion of "correct output" that the simulator needs to enforce on the (aborting) adversary. This is also true for any randomized inputless functionality, and indeed for this reason, our result extends to such functionalities. Note, however, that this is not true for general functionalities where each party has an input.

Non-Aborting Case. We next consider the case where the adversary does not abort in the third round with noticeable probability. Note that in this case, when one execution thread completes, the simulator learns the random strings r_j committed to by the adversarial parties by simply observing the adversary's message in the third round.

At this point, the simulator queries the ideal functionality to obtain the random output (say) R and then attempts to "force" it on the adversary. This involves simulating the simulation-extractable promise ZK and sending a "programmed" value r_i' on behalf of one of the honest parties so that it leads to the desired output R. Now, since the adversary does not abort in the last round with noticeable probability, it would seem that after a polynomial number of trials, the simulator should succeed in forcing the output. At this point, it seems that we have successfully constructed a three round multiparty coin-tossing protocol, which contradicts the lower bound of [GMPP16]!

We now explain the flaw in the above argument. Note that an adversary's aborting behavior may depend upon the output it receives in the last round. For example, it may always choose to abort if it receives an output that starts with 00. Thus, if the simulator attempts to repeatedly force the same random output on the adversary, it may never succeed.

This is where list coin-tossing comes into the picture. In list coin-tossing, the simulator obtains a polynomial number of random strings from the ideal functionality, as opposed to a single string in regular coin-tossing. Our simulator attempts to force each of (polynomially many) random strings one-by-one on the adversary, in the manner as explained above. Now, each of the trials are independent, and therefore the simulator is guaranteed to succeed in forcing one of the random strings after a polynomial number of attempts.

Organization. We define some preliminaries in Section 3 and some building blocks for our protocols in Section 4. In Section 5, we give the definition and construction of Promise ZK. This is followed by the construction and proof of our four round maliciously secure MPC protocol in Section 6.

Then, in Section 7, we give the definition and construction of Simulation Extractable Promise ZK. This is followed by the definition, construction and proof of List Coin Tossing in Section 8.

3 Preliminaries

Here, we recall some preliminaries that will be useful in the rest of the paper. Throughout this paper, we will use λ to denote the security parameter, and $\mathsf{negl}(\lambda)$ to denote any function that is asymptotically smaller than $\frac{1}{\mathsf{poly}(\lambda)}$ for any polynomial $\mathsf{poly}(\cdot)$. We will use PPT to describe a probabilistic polynomial time machine. We will also use the words "rounds" and "messages" interchangeably, whenever clear from context.

3.1 Non-Malleable Commitments

We follow the definition of non-malleable commitments by Pass and Rosen [PR05] and further refined by Lin et al [LPV08] and Goyal [Goy11]. (All of these definitions build upon the original definition of Dwork et al. [DDN91]). Further, we consider delayed-input non malleable commitments where the committer chooses his input only in the last round. In the real interaction, there is a man-in-the-middle adversary MIM interacting with a committer $\mathcal{C}$ (where $\mathcal{C}$ commits to value v) in the left session, and interacting with receiver $\mathcal{R}$ in the right session. Prior to the interaction, the value v is given to C as local input. MIM receives an auxiliary input z, which might contain a-priori information about v. Then the commit phase is executed. Let $\mathsf{MIM}_{\langle C,R\rangle}(\mathsf{val}, z)$ denote a random variable that describes the value $\widetilde{\mathsf{val}}$ committed by the MIM in the right session, jointly with the view of the MIM in the full experiment. In the simulated experiment, a PPT simulator $\mathcal{S}$ directly interacts with the MIM. Let $\mathsf{Sim}_{\langle C,R\rangle}(1^\lambda, z)$ denote the random variable describing the value $\widetilde{\mathsf{val}}$ committed to by $\mathcal{S}$ and the output view of $\mathcal{S}$. If the tags in the left and right interaction are equal, the value $\widetilde{\mathsf{val}}$ committed in the right interaction, is defined to be $\perp$ in both experiments. Also, we only consider synchronizing man-in-the-middle adversaries.

Definition 1 (Non-malleable Commitments w.r.t. Commitment)**.** *A commitment scheme $\langle C, R\rangle$ is said to be non-malleable if for every PPT MIM, there exists a PPT simulator $\mathcal{S}$ such that the following ensembles are computationally indistinguishable:*

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

Goyal et al.[GPR16] construct three-round non-malleable commitments from injective OWFs. Also, in their construction, there exists a polynomial time extractor that extracts the value committed to by the MIM via a single rewind with noticeable probability, if MIM honestly completed the main thread with noticeable probability. Since we require the existence of such an extractor, we cannot rely on the two-message protocols that achieve non-malleability with respect to opening [GKS16].

3.2 Secure Multiparty Computation

As in [Gol04], we follow the real-ideal paradigm for defining secure multi-party computation. The only difference is that our simulator can run in super-polynomial time. A formal definition can be found in Appendix A.

Semi-malicious adversary: An adversary is said to be semi-malicious if it follows the protocol correctly, but with potentially maliciously chosen randomness. We refer the reader to Appendix A for more details.

4 Bulding Blocks

We now describe some of the building blocks we use in our constructions.

4.1 Trapdoor Generation Protocol

In this section, we define and construct a primitive called Trapdoor Generation Protocol. In such a protocol, a sender S (a.k.a. trapdoor generator) communicates with a receiver R. The protocol satisfies two properties: (i) Sender security, i.e., no cheating PPT receiver can learn the trapdoor, and (ii) Extraction, i.e., there exists an expected PPT algorithm (a.k.a. extractor) that can extract the trapdoor from an adversarial sender via rewinding.

Three-round trapdoor generation protocols are known in the literature based on various assumptions [PRS02, Ros04, COSV17a]. Here, we consider trapdoor generation protocols with a stronger sender security requirement that we refer to as *1-rewinding security*. Below, we formally define this notion and then proceed to give a three-round construction for the same based on one-way functions. Our construction is a minor variant of the trapdoor generation protocol from [COSV17a].

Syntax. A trapdoor generation protocol $\mathsf{TDGen} = (\mathsf{TDGen}_1, \mathsf{TDGen}_2, \mathsf{TDGen}_3, \mathsf{TDOut}, \mathsf{TDValid})$ is a 3 round protocol between two parties - a sender (trapdoor generator) S and receiver R that proceeds as below. Additionally, the protocol also has an associated algorithm TDExt which helps to extract a valid trapdoor from the protocol messages.

1. **Round 1 -** $\mathsf{TDGen}_1(\cdot)$**:**
   S computes and sends $\mathsf{td}_1^{S\to R} \leftarrow \mathsf{TDGen}_1(\mathsf{r}_S)$ using a random string r_S.
2. **Round 2 -** $\mathsf{TDGen}_2(\cdot)$**:**
   R computes and sends $\mathsf{td}_2^{R\to S} \leftarrow \mathsf{TDGen}_2(\mathsf{td}_1^{S\to R})$ using randomness r_R.
3. **Round 3 -** $\mathsf{TDGen}_3(\cdot)$**:**
   S computes and sends $\mathsf{td}_3^{S\to R} \leftarrow \mathsf{TDGen}_3(\mathsf{td}_1^{S\to R}, \mathsf{td}_2^{R\to S}; \mathsf{r}_S)$

4. **Output -** $\mathsf{TDOut}(\cdot)$
   The receiver R outputs $\mathsf{TDOut}(\mathsf{td}_1^{S\to R}, \mathsf{td}_2^{R\to S}, \mathsf{td}_3^{S\to R})$.

5. **Trapdoor Validation Algorithm -** $\mathsf{TDValid}(\cdot)$**:**
   Given input $(\mathsf{t}, \mathsf{td}_1^{S\to R})$, output a single bit 0 or 1 that determines whether the value t is a valid trapdoor corresponding to the message td_1 sent in the first round of the trapdoor generation protocol.

Note that the algorithm $\mathsf{TDValid}$ does not form a part of the interaction between the trapdoor generator and the receiver. It is, in fact, a public algorithm that enables publication verification of whether a value t is a valid trapdoor for a first round message td_1 generated by the trapdoor generator.

Extraction. There exists an expected PPT extractor TDExt that, given a set of values $(\mathsf{td}_1, \{\mathsf{td}_2^i, \mathsf{td}_3^i\}_{i=1}^3)$ such that $\mathsf{TDOut}(\mathsf{td}_1, \mathsf{td}_2^i, \mathsf{td}_3^i) = 1$ for all $i \in [3]$, outputs a trapdoor t such that $\mathsf{TDValid}(\mathsf{t}, \mathsf{td}_1) = 1$.

1-Rewinding Security. We define the notion of *1-rewinding security* for a trapdoor generation protocol TDGen. Consider the following experiment between a sender S and a receiver R.

Experiment E:

- R interacts with S and completes one execution of the protocol TDGen. R receives values $(\mathsf{td}_1, \mathsf{td}_3)$ in rounds 1 and 3 respectively.
- Then, R rewinds S to the beginning of round 2.
- R sends S a new second round message td_2^* and receives a message td_3^* in the third round.
- At the end of the experiment, R outputs a value t^*.

Definition 2 (1-Rewinding Security)**.** *A trapdoor generation protocol* $\mathsf{TDGen} = (\mathsf{TDGen}_1, \mathsf{TDGen}_2, \mathsf{TDGen}_3, \mathsf{TDOut}, \mathsf{TDValid})$ *achieves 1-rewinding security if, for every non-uniform PPT receiver* R^* *in the above experiment E,*

$$\Pr\left[\mathsf{TDValid}(\mathsf{t}^*, \mathsf{td}_1) = 1\right] \leq \mathsf{negl}(\lambda),$$

where the probability is over the random coins of V*.* t^* *is the output of* R^* *in the experiment* E *and* td_1 *is the message from* S *in round 1.*

4.1.1 Construction

We now construct a three round trapdoor generation protocol assuming the existence of one way functions.

Consider a sender S and a receiver R who wish to run the trapdoor generation protocol. Let λ denote the security parameter. Let $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be an unforgeable signature scheme based on one-way functions.

Theorem 5. *Assuming the existence of one way functions, the protocol* π^{TD} *described in Figure 1 is a trapdoor generation protocol that satisfies the notion of 1-rewinding security.*

1. **Round 1 -** $\mathsf{TDGen}_1(\mathsf{r}_S)$**:**
   S does the following:
   - Generate $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(\mathsf{r}_S)$.
   - Send $\mathsf{td}_1^{S\to R} = (\mathsf{vk})$ to R.
2. **Round 2 -** $\mathsf{TDGen}_2(\mathsf{td}_1^{S\to R})$**:**
   R sends a random string m as the message $\mathsf{td}_2^{R\to S}$ to S.
3. **Round 3 -** $\mathsf{TDGen}_3(\mathsf{td}_1^{S\to R}, \mathsf{td}_2^{R\to S}; \mathsf{r}_S)$**:**
   S computes and sends $\mathsf{td}_3^{S\to R} = \mathsf{Sign}(\mathsf{sk}, \mathsf{m}; \mathsf{r}_\mathsf{m})$ where r_m is randomly picked.
4. **Output: -** $\mathsf{TDOut}(\mathsf{td}_1^{S\to R}, \mathsf{td}_2^{R\to S}, \mathsf{td}_3^{S\to R})$
   The receiver R outputs 1 if $\mathsf{Verify}(\mathsf{td}_1^{S\to R}, \mathsf{m}, \mathsf{td}_3^{S\to R}) = 1$.
5. **Trapdoor Validation Algorithm -** $\mathsf{TDValid}(\mathsf{t}, \mathsf{td}_1)$**:**
   Given input $(\mathsf{t}, \mathsf{td}_1)$, the algorithm does the following:
   - Let $\mathsf{t} = (\mathsf{m}_i, \sigma_i)_{i=1}^3$.
   - Output 1 if $\mathsf{Verify}(\mathsf{td}_1, \mathsf{m}_i, \sigma_i) = 1$ for all $i \in [3]$.

Figure 1: Trapdoor Generation Protocol.

Proof. Suppose the protocol π^{TD} is not 1-rewinding secure. That is, there exists a malicious receiver R^* that breaks the 1-rewinding security. We will use R^* to design an adversary $\mathcal{A}_{\mathsf{Sign}}$ that breaks the unforgeability of the signature scheme. In the experiment E, $\mathcal{A}_{\mathsf{Sign}}$ performs the role of S and interacts with R^*. Also, $\mathcal{A}_{\mathsf{Sign}}$ interacts with a challenger $\mathcal{C}_{\mathsf{Sign}}$ and receives a verification key vk which is set as td_1 and sent to R^* in round 1.

Now, on receiving a query $\mathsf{td}_2 = \mathsf{m}$ from R^*, $\mathcal{A}_{\mathsf{Sign}}$ forwards this to $\mathcal{C}_{\mathsf{Sign}}$ and receives a value σ_m from $\mathcal{C}_{\mathsf{Sign}}$ which it sends to R^* as the message td_3. Then, on receiving a query $\mathsf{td}_2^* = \mathsf{m}^*$ from R^* in the rewound execution, $\mathcal{A}_{\mathsf{Sign}}$ once again does the same. That is, $\mathcal{A}_{\mathsf{Sign}}$ forwards this to $\mathcal{C}_{\mathsf{Sign}}$ and receives a value σ_{m^*} from $\mathcal{C}_{\mathsf{Sign}}$ which it sends to R^* as the message td_3^*.

Then, since R^* breaks the 1-rewinding security, it outputs a value t^* in experiment E such that $\mathsf{TDValid}(\mathsf{t}^*, \mathsf{td}_1) = 1$ with non-negligible probability p. Recall from the definition of the algorithm $\mathsf{TDValid}$, it must be the case that $\mathsf{t}^* = \{\mathsf{m}_i, \sigma_i\}_{i=1}^3$ such that $\mathsf{Verify}(\mathsf{vk}, \mathsf{m}_i, \sigma_i) = 1$ for all i. $\mathcal{A}_{\mathsf{Sign}}$ picks the value $\mathsf{m}_i \notin \{\mathsf{m}, \mathsf{m}^*\}$ and outputs (m_i, σ_i) to the challenger $\mathcal{C}_{\mathsf{Hid}}$ as a forgery. □

Extractor $\mathsf{TDExt}(\cdot)$**.** The extractor works as follows. It receives a verification key $\mathsf{vk} = \mathsf{td}_1$, and a set of values (m_i, σ_i) for all $i \in [3]$, such that $\mathsf{Verify}(\mathsf{vk}, \mathsf{m}_i, \sigma_i) = 1$. Then, TDExt outputs $\mathsf{t} = (\mathsf{m}_i, \sigma_i)_{i=1}^3$ as a valid trapdoor. Correctness of the extraction is easy to see by inspection.

4.2 WI with Bounded Rewinding Security

We define the notion of 3 round delayed input witness indistinguishable argument with "Bounded Rewinding" security and construct such a primitive assuming the existence of injective one way functions. Such a primitive has been implicitly constructed and used in literature previously. For example, in Goyal et al. [GRRV14], they construct and use the notion of WI arguments with 1-rewinding security based on injective one way functions. For the sake of completeness we formally

define it here and briefly describe the construction in Appendix B. In our applications, we will instantiate the rewinding parameter L with the value 5.

Definition 3 (3 Round Delayed Input WI with Bounded Rewinding Security)**.** *Fix a positive integer L. A delayed input witness indistinguishable argument system* $\mathsf{RWI} = (\mathsf{Prove}, \mathsf{Valid})$ *(a formal definition of it can be found in [JKKR17]) is said to satisfy L-Rewinding Security if for every non-uniform PPT verifier V^*, there exists a simulator $\mathcal{S}$ such that the output of the experiments* REAL *and* IDEAL *defined below are indistinguishable.*

Experiment REAL***:***

1. *V^* interacts with an honest prover P that has access to an efficiently sampleable distribution of (instance, witness) pairs. V^* completes one execution of the protocol* RWI *and receives values* $(\mathsf{rwi}_1, \mathsf{rwi}_3)$ *in rounds 1 and 3 respectively.*
2. *Then, V^* rewinds P to the beginning of round 2 and sets a counter value to 0.*
3. *V^* sends P a new second round message* rwi_2^* *and receives a message* rwi_3^* *in the third round.*
4. *V^* updates the value of the counter. If counter $< L$, go back to step 2.*
5. *At the end of the experiment, V^* outputs some value.*

Experiment IDEAL***:***

1. *V^* interacts with a simulator* Sim *that is able to sample only instances using the same distribution as the honest prover P. V^* completes one execution of the protocol* RWI *and receives values* $(\mathsf{rwi}_1, \mathsf{rwi}_3)$ *in rounds 1 and 3 respectively.*
2. *Then, V^* rewinds P to the beginning of round 2 and sets a counter value to 0.*
3. *V^* sends P a new second round message* rwi_2^* *and receives a message* rwi_3^* *in the third round.*
4. *V^* updates the value of the counter. If counter $< L$, go back to step 2.*
5. *At the end of the experiment, V^* outputs some value.*

4.3 Extractable Commitment with Bounded Rewinding Security - BRew.ECom

In this section, we describe a three round delayed-input extractable commitment protocol that achieves hiding property against an a priori bounded number of rewinds. Consider a committer C with input x and a receiver R. We consider hiding of the commitment scheme against a malicious receiver who can rewind the committer back to the start of the second round an a priori fixed bounded number of times (K times as defined in the protocol) and the security property we require is that the committed value still remains hidden. However, unlike the "1-rewinding security" property of the trapdoor generation protocol, we don't explicitly define or prove a K- rewinding security property for this extractable commitment scheme. Instead, this is done inline in the application - the four round MPC protocol (in Section 6) by opening up the details of the construction to make the exposition easier. In our particular application, we set the value of the parameter K to be 4. However, in general, K can be any polynomial in the security parameter λ.

In the next subsection, we use this primitive to build another extractable commitment scheme which is then used in the construction of our 4 round MPC protocol.

Construction. We use a non-interactive commitment scheme Com to build the scheme. The protocol $(\mathsf{BRew.ECom}_1, \mathsf{BRew.ECom}_2, \mathsf{BRew.ECom}_3)$ is described in Figure 2. We also define a property called "well-formedness" of the commitment scheme. Additionally, the scheme $\mathsf{BRew.ECom}$ has an associated algorithm $\mathsf{Ext}_{\mathsf{BRew.ECom}}$. The property we require from the extractor $\mathsf{Ext}_{\mathsf{BRew.ECom}}$ is that given a set of $(\mathsf{K}+1)$ "well-formed" executions of the commitment scheme using the same sender message for the first round, the extractor successfully extracts the message inside the commitment except with negligible probability. The description of the extractor $\mathsf{Ext}_{\mathsf{BRew.ECom}}$ follows after the scheme.

Sender S has input x.

Commitment Phase:

1. **Round 1:**
   S does the following:
   - Pick λ random degree K polynomials $\mathsf{p}_1, \ldots, \mathsf{p}_\lambda$.
   - Compute $\mathsf{brew.ec}_{1,l}^{S\to R} = \mathsf{Com}(\mathsf{p}_l)$ for all $l \in [\lambda]$.
   - Send $\mathsf{brew.ec}_1^{S\to R} = (\mathsf{brew.ec}_{1,1}^{S\to R}, \ldots, \mathsf{brew.ec}_{1,\lambda}^{S\to R})$ to R.
2. **Round 2:**
   R does the following:
   - Pick random values $\mathsf{z}_1, \ldots, \mathsf{z}_\lambda$.
   - Send $\mathsf{brew.ec}_2^{R\to S} = (\mathsf{z}_1, \ldots, \mathsf{z}_\lambda)$ to S.
3. **Round 3:**
   S does the following:
   - Compute $\mathsf{brew.ec}_{3,l}^{S\to R} = (x \oplus \mathsf{p}_l(0), \mathsf{p}_l(\mathsf{z}_l))$ for all $l \in [\lambda]$.
   - Send $\mathsf{brew.ec}_3^{S\to R} = (\mathsf{brew.ec}_{3,1}^{S\to R}, \ldots, \mathsf{brew.ec}_{3,\lambda}^{S\to R})$ to R.

Decommitment Phase:

1. S outputs $\mathsf{p}_1, \ldots, \mathsf{p}_\lambda$ along with the randomness used in the first round commitments.
2. First, for each l, R verifies the commitment of p_l given in round 1 and the correctness of the value $\mathsf{p}_l(\mathsf{z}_l)$ in round 3.
3. Then, for each l, R computes x using the message $(x \oplus \mathsf{p}_l(0))$ and outputs $\perp$ if x is not the same across all values of l.

Figure 2: Extractable Commitment Scheme $\mathsf{BRew.ECom}$.

Well-formedness: The sender's messages $\mathsf{brew.ec}_1^{S\to R}, \mathsf{brew.ec}_3^{S\to R}$ are said to be well-formed with respect to the receiver's message $\mathsf{brew.ec}_2^{R\to S}$ if:
There exists x and a set A of size $(\lambda - 1)$ such that for every l in this set, the following holds:

- $\mathsf{brew.ec}_{1,l}^{S\to R} = \mathsf{Com}(\mathsf{p}_l)$ (AND)
- $\mathsf{brew.ec}_{3,l}^{S\to R} = (x \oplus \mathsf{p}_l(0), \mathsf{p}_l(\mathsf{z}_l))$, where z_l was received in round 2.

Extractor $\mathsf{Ext}_{\mathsf{BRew.ECom}}$: Recall that what we require from the extractor $\mathsf{Ext}_{\mathsf{BRew.ECom}}$ is that given a set of $(\mathsf{K}+1)$ "well-formed" executions of the commitment scheme using the same sender message for the first round, the extractor successfully extracts the message inside the commitment except with negligible probability. As in the case of the trapdoor generation protocol, note that in reality, there is an expected PPT rewinding procedure to obtain these $(\mathsf{K}+1)$ "well-formed" tuples from the malicious sender. The simulator in our applications does that and then feeds these tuples to the extractor $\mathsf{Ext}_{\mathsf{BRew.ECom}}$.

The extractor's strategy is described in Figure 3.

Input: $(\mathsf{brew.ec}_1, \{\mathsf{brew.ec}_2^i, \mathsf{brew.ec}_3^i\}_{i=1}^{\mathsf{K}+1})$ where $(\mathsf{brew.ec}_1, \mathsf{brew.ec}_3^i)$ denote messages from the sender and $\mathsf{brew.ec}_2^i$ denote the receiver's messages. Also, for all i, $(\mathsf{brew.ec}_1, \mathsf{brew.ec}_2^i, \mathsf{brew.ec}_3^i)$ is "well-formed".

1. For each $j \in [\lambda]$, do:
   - Using the set of values $\{\mathsf{z}_j^i, \mathsf{p}_j(\mathsf{z}_j^i)\}$ for all $i \in [\mathsf{K}+1]$, compute a degree K polynomial p_j.
   - Let $\mathsf{brew.ec}_{3,j}^i = (a_j^i, b_j^i)$.
   - If a_j^i is equal for all i, compute $x_j = (a_j^i \oplus \mathsf{p}_j(0))$.
   - Else, set $x_j = \perp$.
2. Consider the set of values $x_1, \ldots, \mathsf{x}_\lambda$. If majority of these values is equal, output that value x. Else, output $\perp$.

Figure 3: Strategy of algorithm $\mathsf{Ext}_{\mathsf{BRew.ECom}}$.

We now analyze why the extraction is successful. We know that for all $i \in [\mathsf{K}+1]$, the messages $(\mathsf{brew.ec}_1, \mathsf{brew.ec}_2^i, \mathsf{brew.ec}_3^i)$ form a "well-formed" commitment scheme. So, for each i, there exists at most one j such that $\mathsf{brew.ec}_{3,j}^i = (a_j^i, b_j^i)$ was not computed honestly. That is, putting it together, across all $i \in [\mathsf{K}+1]$, there exists at most $(\mathsf{K}+1)$ values of $\mathsf{brew.ec}_{3,j}^i$ that were not computed honestly. This implies that for atleast $(\lambda - \mathsf{K} - 1)$ values of j, the values $\mathsf{brew.ec}_{3,j}^i$ were computed correctly for all i and this completes the proof.

Claim 1. *Assuming the hiding of the commitment scheme* Com*, the above scheme* $\mathsf{BRew.ECom}$ *is hiding.*

Proof. We will now prove this claim via a series of intermediate hybrids Hyb_1 to Hyb_4 where Hyb_1 corresponds to using input x_1 and Hyb_4 corresponds to using input x_2.

- Hyb_1**:** This is the real experiment with sender input x_1.
- Hyb_2**:** In round 1, compute $\mathsf{brew.ec}_{1,l} = \mathsf{Com}(0)$ for all l.
  This is indistinguishable from the previous hybrid by the hiding property of the scheme Com.
- Hyb_3**:** For each l, pick a new degree K polynomial q_l such that $(x_1 \oplus \mathsf{p}_l(0)) = (x_2 \oplus \mathsf{q}_l(0))$. Compute $\mathsf{brew.ec}_{3,l}$ as $(x_2 \oplus \mathsf{q}_l(0), \mathsf{q}_l(\mathsf{z}_l))$.
  This hybrid is statistically indistinguishable from the previous hybrid.

- Hyb_4**:** In round 1, compute $\mathsf{brew.ec}_{1,l} = \mathsf{Com}(\mathsf{q}_l)$ for all l. This corresponds to the real experiment with sender input x_2.
  This is indistinguishable from the previous hybrid by the hiding property of the scheme Com.

□

4.4 Extractable Commitment with Reusability - R.ECom

We use this primitive in the construction of our 4 round MPC protocol. Informally, this is a 3 round delayed input protocol that has some form of reusability property. That is, consider a committer C with input m and a receiver R. Informally, the property we require is that for a fixed input m and fixed messages for the first 2 rounds, the sender can generate several distinct third round messages. Once again, we don't explicitly define or prove the reusability property for this extractable commitment scheme here. Instead, this is done inline in the application - the 4 round MPC protocol by opening up the details of the construction to make the exposition easier. Also, in the 4 round MPC protocol, when we invoke this scheme, we set the value of the parameter K to be 4. However, in general, K can be any polynomial in the security parameter λ.

To build our scheme $\mathsf{R.ECom}$, we use a pseudorandom function PRF and the above extractable commitment scheme $\mathsf{BRew.ECom} = (\mathsf{BRew.ECom}_1, \mathsf{BRew.ECom}_2, \mathsf{BRew.ECom}_3, \mathsf{Ext}_{\mathsf{BRew.ECom}})$. The protocol $(\mathsf{R.ECom}_1, \mathsf{R.ECom}_2, \mathsf{R.ECom}_3)$ is described in Figure 4. As before, we also define a property called "well-formedness" of the commitment scheme.

Additionally, the scheme $\mathsf{R.ECom}$ has an associated algorithm $\mathsf{Ext}_{\mathsf{R.ECom}}$. Once again, the property we require from the extractor $\mathsf{Ext}_{\mathsf{R.ECom}}$ is that given a set of $(\mathsf{K}+1)$ "well-formed" executions of the commitment scheme using the same sender message for the first round, the extractor successfully extracts the message inside the commitment except with negligible probability.

The description of the extractor $\mathsf{Ext}_{\mathsf{R.ECom}}$ follows after the scheme.

Well-formedness: The sender's messages $\mathsf{r.ec}_1^{S\to R}, \mathsf{r.ec}_3^{S\to R}$ are said to be well-formed with respect to the receiver's message $\mathsf{r.ec}_2^{R\to S}$ if the underlying messages $(\mathsf{brew.ec}_1^{S\to R}, \mathsf{brew.ec}_3^{S\to R})$ are well-formed for the scheme $\mathsf{BRew.ECom}$ with respect to the receiver's message $\mathsf{brew.ec}_2^{R\to S}$.

Extractor $\mathsf{Ext}_{\mathsf{R.ECom}}$**:** Recall that what we require from the extractor $\mathsf{Ext}_{\mathsf{R.ECom}}$ is that given a set of $(\mathsf{K}+1)$ "well-formed" executions of the commitment scheme using the same sender message for the first round, the extractor successfully extracts the message inside the commitment except with negligible probability. Once again, note that in reality, there is an expected PPT rewinding procedure to obtain these $(\mathsf{K}+1)$ "well-formed" tuples from the malicious sender. The simulator in our applications does that and then feeds these tuples to the extractor $\mathsf{Ext}_{\mathsf{R.ECom}}$.

$\mathsf{Ext}_{\mathsf{R.ECom}}$ runs the extractor $\mathsf{Ext}_{\mathsf{BRew.ECom}}$ of the underlying extractable commitment scheme $\mathsf{BRew.ECom}$ to recover r. Then, $\mathsf{Ext}_{\mathsf{R.ECom}}$ computes $m = \beta^{S\to R} \oplus \mathsf{PRF}(\mathsf{r}, \alpha^{S\to R})$. Since $\mathsf{Ext}_{\mathsf{BRew.ECom}}$ is a successful PPT extractor given a set of $(\mathsf{K}+1)$ executions of "well-formed" commitment messages, the same applies to $\mathsf{Ext}_{\mathsf{R.ECom}}$ as well.

Claim 2. *Assuming the security of the extractable commitment* $\mathsf{BRew.ECom}$ *and the pseudorandom function* PRF*, the above scheme* $\mathsf{R.ECom}$ *is hiding.*

Proof. We will now prove this claim via a series of intermediate hybrids Hyb_1 to Hyb_5 where Hyb_1 corresponds to using input m_1 and Hyb_5 corresponds to using input m_2.

- Hyb_1**:** This is the real experiment with sender input m_1.

Sender S has input m.

Commitment Phase:

1. **Round 1:**
   S does the following:
   - Compute $\mathsf{brew.ec}_1^{S\to R} \leftarrow \mathsf{BRew.ECom}_1(\mathsf{r}_{\mathsf{brew.ec}})$ using randomness $\mathsf{r}_{\mathsf{brew.ec}}$.
   - Send $\mathsf{r.ec}_1^{S\to R} = \mathsf{brew.ec}_1^{S\to R}$ to R.
2. **Round 2:**
   R does the following:
   - Compute $\mathsf{brew.ec}_2^{R\to S} \leftarrow \mathsf{BRew.ECom}_2(\mathsf{brew.ec}_1^{S\to S})$ using some randomness.
   - Send $\mathsf{r.ec}_2^{R\to S} = \mathsf{brew.ec}_2^{R\to S}$ to S.
3. **Round 3:**
   S does the following:
   - Pick two random strings $\alpha^{S\to R}, \mathsf{r}$.
   - Compute $\mathsf{brew.ec}_3^{S\to R} \leftarrow \mathsf{BRew.ECom}_3(\mathsf{r}, \mathsf{brew.ec}_1^{S\to R}, \mathsf{brew.ec}_2^{R\to S}; \mathsf{r}_{\mathsf{brew.ec}})$.
   - Compute $\beta^{S\to R} = \mathsf{PRF}(\mathsf{r}, \alpha^{S\to R}) \oplus m$.
   - Send $\mathsf{r.ec}_3^{S\to R} = (\alpha^{S\to R}, \mathsf{brew.ec}_3^{S\to R}, \beta^{S\to R})$ to R.

Decommitment Phase:

1. S outputs the decommitment of the underlying scheme $\mathsf{BRew.ECom}$.
2. R runs the decommitment phase of $\mathsf{BRew.ECom}$ to recover r.
3. Then, R computes $m = \mathsf{PRF}(\mathsf{r}, \alpha^{S\to R}) \oplus \beta^{S\to R}$.

Figure 4: Extractable Commitment Scheme $\mathsf{R.ECom}$.

- Hyb_2**:** In this hybrid, pick a random value γ and compute $\mathsf{brew.ec}_3^{S\to R} = \mathsf{BRew.ECom}_3(\gamma, \mathsf{brew.ec}_1^{S\to R}, \mathsf{brew.ec}_2^{R\to S}; \mathsf{r}_{\mathsf{brew.ec}})$.
  This is indistinguishable from the previous hybrid by the hiding property of the underlying extractable commitment scheme $\mathsf{BRew.ECom}$.
- Hyb_3**:** In this hybrid, compute $\beta^{S\to R}$ uniformly at random.
  This is indistinguishable from the previous hybrid by the security of the pseudorandom function PRF.
- Hyb_4**:** In this hybrid, compute $\beta^{S\to R} = \mathsf{PRF}(\mathsf{r}, \alpha^{S\to R}) \oplus m_2$.
  This is indistinguishable from the previous hybrid by the security of the pseudorandom function PRF.
- Hyb_5**:** In this hybrid, compute $\mathsf{brew.ec}_3^{S\to R} = \mathsf{BRew.ECom}_3(\mathsf{r}, \mathsf{brew.ec}_{a,1}^{S\to R}, \mathsf{brew.ec}_2^{R\to S}; \mathsf{r}_{\mathsf{brew.ec}})$.
  This corresponds to the real experiment with sender input m_2.
  This is indistinguishable from the previous hybrid by the hiding property of the underlying

extractable commitment scheme $\mathsf{BRew.ECom}$.

□

5 Promise Zero Knowledge

In this section, we introduce a new notion of promise zero knowledge interactive arguments. Unlike the standard notion of zero knowledge interactive arguments that is defined in the unidirectional-message model of communication, promise ZK is defined in the simultaneous-message model, where in every round, both the prover and the verifier simultaneously send a message to each other. Crucially, in promise ZK, the zero knowledge property is only required to hold against a specific class of non-aborting verifiers.

We start by formalizing interactive arguments in the simultaneous-message model (Section 5.1). We define our new notion of promise ZK (Section 5.2). Later, in Section 7, we define the notion of simulation-extractable promise ZK.

5.1 Simultaneous-Message Interactive Arguments

We formalize the notion of interactive arguments in the simultaneous-message model, or simultaneous-message interactive arguments in short. In fact, we will consider simultaneous-message interactive arguments in the delayed-input setting where the parties receive their inputs in the last round of the protocol.

Simultaneous-Message Interactive Protocols. In an interactive protocol in the simultaneous-message model (or in short, a simultaneous-message interactive protocol), in every round (including the last), both parties send a message simultaneously to each other. However, a rushing adversary, who corrupts either prover or verifier, may wait to receive the message of the other party in a round, before sending its own message in that round.

Delayed-Input Interactive Protocols. We will focus on simultaneous-message interactive protocols for deciding languages. We start by describing some notation and terminology. A protocol execution between a prover P and a verifier V with instance x and witness w is denoted as $\langle P, V\rangle(x, w)$. Whenever clear from context, we also use the same notation to denote the output of V. We use $(\mathsf{Prove}, \mathsf{Verify})$ to denote the algorithms used by P and V respectively. Let $\mathsf{Trans}(P, V)$ denote the transcript of the execution between the parties P and V.

An n-round delayed-input simultaneous-message interactive protocol $(\mathsf{Prove}, \mathsf{Verify})$ between P and V for deciding a language L with associated relation R_L proceeds in the following manner:

- At the beginning of the protocol, P and V receive the size of the instance and execute the first $(n-2)$ rounds.
- At the start of the second last round, P receives an input $(x = (x_1, x_2), w) \in R_L$ and V receives x_1.
- In the last round, V receives the other part of the statement x_2. Upon receiving the last round message from P , V outputs 1 or 0.

Looking ahead, we consider such a notion of delayed-input because in both our applications (the four round MPC protocol and the three round protocols), the statement is in fact split into two parts - one decided in the second last round and the other part decided only in the last round.

In the sequel, we will consider two kinds of adversaries for delayed-input protocols:

- *Adaptive Provers*: We will consider *adaptive* adversarial provers, who may choose the instance in an adaptive manner, depending upon the partial transcript of the protocol.
- *Non-adaptive Verifiers*: We will consider adversarial verifiers who receive the instance in the same manner as an honest verifier (i.e., they receive the first part of the statement in the second last round and the second part of the statement in the last round). We refer to such adversarial verifiers as *non-adaptive*. As we will see later, security against non-adaptive verifiers suffices for our applications of promise zero-knowledge.

Delayed-Input Interactive Arguments. An n-round delayed-input simultaneous-message interactive argument for a language L must satisfy the standard notion of completeness as well as soundness against adaptive provers.

Definition 4 (Delayed-Input Simultaneous-Message Interactive Arguments)**.** *An n-round delayed-input interactive protocol* $(\mathsf{Prove}, \mathsf{Verify})$ *between P and V in the simultaneous-message model for deciding a language L is an argument system for L if it satisfies the following properties:*

- **Completeness:** *For every $(x, w) \in R_L$,*

$$\Pr\big[\langle P, V\rangle(x, w) = 1\big] \geq 1 - \mathsf{negl}(\lambda),$$

  where the probability is over the random coins of P and V.

- **Adaptive Soundness:** *For every $z \in \{0,1\}^*$, every PPT prover P^* that chooses $x \in \{0,1\}^\lambda \setminus L$ adaptively,*

$$\Pr\big[\langle P^*(z), V\rangle(x) = 1\big] \leq \mathsf{negl}(\lambda),$$

  where the probability is over the random coins of V.

We enhance the syntax for simultaneous-message interactive arguments to include an additional algorithm Valid. That is, a simultaneous-message interactive argument is now denoted as $(\mathsf{Prove}, \mathsf{Verify}, \mathsf{Valid})$. The notions of completeness and soundness remain intact as before. Looking ahead, the intuition behind introducing the new algorithm is that we want to capture those verifiers who send a "valid", non-aborting message in every round (including the last round). We do this by using the Valid algorithm. This algorithm may be different for each protocol. Looking ahead, in our protocols, the prover, at the end of the execution can check whether the verifier sent "valid" messages with noticeable probability and we will require simulation only in such scenarios.

5.2 Promise ZK Definition

We now proceed to describe our notion of promise zero knowledge. Roughly speaking, we define promise ZK similarly to standard ZK, with two notable differences: first, promise ZK is defined in the simultaneous-message model. Further, Zero knowledge is only defined w.r.t. a special class of verifiers who promise to not abort with some noticeable probability.

Before describing the notion, we first define what it means for an algorithm $\mathsf{pExtract}$ to be "Admissible".

5.2.1 Admissible $\mathsf{pExtract}_c$

Consider a delayed-input simultaneous message interactive argument system $(\mathsf{Prove}, \mathsf{Verify}, \mathsf{Valid})$, and consider any verifier V^*. Let $\mathsf{Prove} = (P_1, P_2)$ where $(\mathsf{msg}, \mathsf{st}) \leftarrow P_1(1^\lambda)$, and $P_2(x, w, \mathsf{st})$ continues the rest of the Prove algorithm with V^*. Let $\mathsf{Trans}(P_2(x, w, \mathsf{st}), V^*)$ denote the protocol transcript between P_2 and V^*: that is, $\mathsf{Trans}(P, V^*) = (\mathsf{msg}, \mathsf{Trans}(P_2(x, w, \mathsf{st}), V^*))$.

Let $q_{\mathsf{msg}} = \Pr_{(x,w \leftarrow X,W)}[\mathsf{Valid}(\mathsf{msg}, \mathsf{Trans}(P_2(x, w, \mathsf{st}), V^*)) = 1 | (\mathsf{msg}, \cdot) \leftarrow P_1(1^\lambda)]$ where the probability is over the random choices of (x, w) and the coins of the parties P_2, V^*.

An oracle algorithm $\mathsf{pExtract}_c$ is said to be admissible if the following is true for all malicious verifiers V^* :

- If $\mathsf{pExtract}_c^{V^*}(\mathsf{msg}, \mathsf{st}) = 0$, then $q_{\mathsf{msg}} < 2 \cdot \lambda^{-c}$.
- Otherwise, if $\mathsf{pExtract}_c^{V^*}(\mathsf{msg}, \mathsf{st}) = p$, then $p \geq \lambda^{-c}$ and $\frac{p}{2} < q_{\mathsf{msg}} < 2 \cdot p$.

5.2.2 Promise ZK

We now formalize our notion of promise ZK. We note that this only considers the delayed-input distributional setting.

Definition 5 (Promise Zero Knowledge)**.** *An n-round distributional delayed-input simultaneous-message interactive argument* $(\mathsf{Prove}, \mathsf{Verify}, \mathsf{Valid})$ *where* $\mathsf{Prove} = (P_1, P_2)$ *for a language L is said to be* promise zero knowledge against non-adaptive verifiers *if for every efficiently sampleable distribution $(\mathcal{X}_\lambda, \mathcal{W}_\lambda)$ on R_L, i.e.,* $\mathsf{Supp}(\mathcal{X}_\lambda, \mathcal{W}_\lambda) = \{(x, w) : x \in L \cap \{0,1\}^\lambda, w \in R_L(x)\}$,

every non-adaptive PPT verifier V^*, every $z \in \{0,1\}^*$, every $c > 0$, and all admissible $\mathsf{pExtract}_c$*, there exists a simulator $\mathcal{S} = (P_1, \mathcal{S}_2)$ such that for every PPT distinguisher $\mathcal{D}$:*

Consider the experiments REAL *and* IDEAL *defined below. Let* $\mathsf{View}_{V^*}[\mathsf{REAL}]$ *denote the output of the experiment* REAL *and* $\mathsf{View}_{V^*}[\mathsf{IDEAL}]$ *denote the output of the experiment* IDEAL*. Then:*

1. *The running time of oracle algorithm $\mathcal{S}_2$ on input (z, x, st, p) is $\mathsf{poly}(\lambda) \cdot O(\frac{1}{p})$.*
2. *Then,*
$$\left| \Pr\left[\mathcal{D}(z, \mathsf{View}_{V^*}[\mathsf{REAL}] = 1\right] - \Pr\left[\mathcal{D}(z, \mathsf{View}_{V^*}[\mathsf{IDEAL}]) = 1\right] \right| \leq \lambda^{-c}$$
where the probability is over the random coins of the parties in the below experiments.

Experiment REAL**:** .

- *Compute* $(\mathsf{msg}, \mathsf{st}) \leftarrow P_1(1^\lambda)$*. Output* msg*.*
- *Then, sample* $(x, w) \leftarrow (\mathcal{X}_\lambda, \mathcal{W}_\lambda)$*.*
- *Output* $\langle P_2(x, w, \mathsf{st}), V^* \rangle$*.*

Experiment IDEAL**:** .

- *Compute* $(\mathsf{msg}, \mathsf{st}) \leftarrow P_1(1^\lambda)$. *Output* msg.
- *Let* $\mathsf{pExtract}_c^{V^*}(\mathsf{msg}, \mathsf{st}) = p$.
- *If* $p = 0$, *sample* $(x, w) \leftarrow (\mathcal{X}_\lambda, \mathcal{W}_\lambda)$ *and output* $\langle P_2(x, w, \mathsf{st}), V^* \rangle$
- *Else, sample* $x \leftarrow (\mathcal{X}_\lambda)$ *and output* $\mathcal{S}_2^{V^*}(z, x, \mathsf{st}, p)$.

Going forward, we use *promise ZK argument* to refer to a delayed-input distributional promise zero-knowledge simultaneous-message argument system against non-adaptive verifiers.

5.3 Construction

In this section, we construct a three round Promise ZK argument system for NP. Formally, we prove the following theorem:

Theorem 6. *Assuming the existence of polynomially secure injective one way functions, the protocol* π^{PZK} *is a three round Promise ZK argument.*

We start by describing some notation and cryptographic primitives that we use in our construction.

Building Blocks. Our construction relies on the following list of cryptographic primitives.

- $\mathsf{TDGen} = (\mathsf{TDGen}_1, \mathsf{TDGen}_2, \mathsf{TDGen}_3)$ is the three-message trapdoor generation protocol as defined in Section 4. TDGen also has two associated PPT algorithms $\mathsf{TDValid}, \mathsf{TDExt}$.
  Recall that the algorithm $\mathsf{TDValid}$ takes as input a tuple of 4 values : the first three being outputs of the algorithms $\mathsf{TDGen}_1, \mathsf{TDGen}_2, \mathsf{TDGen}_3$ and outputs 1 if the fourth value is a valid trapdoor with respect to these three.
  The algorithm TDExt, as defined earlier, with overwhelming probability, outputs a valid trapdoor given the transcript of 3 executions of the protocol.
- $\mathsf{RWI} = (\mathsf{RWI}_1, \mathsf{RWI}_2, \mathsf{RWI}_3, \mathsf{RWI}_4)$ is the three round delayed-input witness indistinguishable argument with bounded rewinding security defined in Section 4. The fourth algorithm RWI_4 is the final verification algorithm. We will set the rewinding security parameter L to be 5 in all our applications.
- $\mathsf{PZK.ECom} = (\mathsf{PZK.ECom}_1, \mathsf{PZK.ECom}_2, \mathsf{PZK.ECom}_3)$ is any three-message delayed-input extractable commitment scheme in which the third round message is indistinguishable from a random string when the input is $\perp$. Further, let $\mathsf{Ext}_{\mathsf{PZK.ECom}}$ denote the polynomial time extractor of this scheme. We require that given the transcript of 2 executions of the protocol, $\mathsf{Ext}_{\mathsf{PZK.ECom}}$ can extract the value committed to inside the commitment with non-negligible probability. The extractable commitment schemes defined in [PRS02, Ros04, ACJ17] are a few examples of such a scheme that can be based on one way functions.

NP Languages. Given any NP language L characterized by relation R, we define a new NP language L_{RWI} characterized by the following relation R_{RWI}. This language will be used in our construction in the next subsection.
Statement: $\mathsf{st}_{\mathsf{RWI}} = (x, \mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3, \mathsf{td}_1, \mathsf{td}_2, \mathsf{td}_3)$
Witness: $\mathsf{w}_{\mathsf{RWI}} = (w, \mathsf{t}, \mathsf{r}_\mathsf{c})$
Relation: $R(\mathsf{st}_{\mathsf{RWI}}, \mathsf{w}_{\mathsf{RWI}}) = 1$ if and only if :

- $R(x, w) = 1$

  (OR)

- $\mathsf{TDValid}(\mathsf{td}_1, \mathsf{td}_2, \mathsf{td}_3, \mathsf{t}) = 1$ AND
- $\mathsf{c}_1 = \mathsf{PZK.ECom}_1(\mathsf{r}_\mathsf{c})$ AND
- $\mathsf{c}_3 = \mathsf{PZK.ECom}_3(\mathsf{t}, \mathsf{c}_1, \mathsf{c}_2; \mathsf{r}_\mathsf{c})$.

That is, either :

1. x is in the language L with witness w (OR)
2. $(\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3)$ form a non-malleable commitment to a value t that is a valid trapdoor for the messages $(\mathsf{td}_1, \mathsf{td}_2, \mathsf{td}_3)$ generated using the trapdoor generation algorithms.

5.3.1 The Protocol

Let P denote the prover and V denote the verifier. Let L be any NP language with an associated relation R_L. Let $(\mathcal{X}_\lambda, \mathcal{W}_\lambda)$ be any efficiently sampleable distribution on R_L.

We construct a three round protocol $\pi^{\mathsf{PZK}} = (P, V, \mathsf{Valid})$ for L. The protocol is described in Figure 5. We first describe the algorithms (P, V) denoting the interaction between the prover and verifier. The description of algorithm Valid is given at the end. We use the notation $P \to V$ in the superscript to denote that the message was sent by P to V. The round number of any sub-protocol being used (the trapdoor generation or the bounded rewinding secure WI argument) is written in the subscript.

Completeness follows from the correctness of the bounded rewinding secure WI protocol described in Section 4.

5.4 Security Proof

We will now describe the proof of Soundness and Distributional Promise ZK to complete the proof of Theorem 6.

5.4.1 Soundness

We will prove this by contradiction. Assume that soundness doesn't hold. That is, there exists a cheating prover P^* such that for some statement $x^* \notin L$ of its choice,

$$\Pr\big[\langle P^*(z), V\rangle(x) = 1\big] \geq p,$$

where p is a non-negligible function and the probability is over the random coins of V. Then, assuming the soundness of the bounded rewinding secure WI argument system, we will use this

Inputs: Prover P has input $(\mathcal{X}_\lambda, \mathcal{W}_\lambda)$ - an efficiently sampleable distribution on R_L.

1. **Round 1:**
   - **Prover message:**
     Compute and send $(\mathsf{rwi}_1^{P\to V}) \leftarrow \mathsf{RWI}_1(1^\lambda)$, $\mathsf{c}_1^{P\to V} \leftarrow \mathsf{PZK.ECom}_1(\mathsf{r_c})$ using a random string $\mathsf{r_c}$.
   - **Verifier message:**
     Compute and send $\mathsf{td}_1^{V\to P} \leftarrow \mathsf{TDGen}_1(\mathsf{r_{td}})$ using a random string $\mathsf{r_{td}}$.
2. **Round 2:**
   - **Prover message:**
     - Sample $(x = (x_1, x_2), w) \leftarrow (\mathcal{X}_\lambda, \mathcal{W}_\lambda)$.
     - Compute and send $\mathsf{td}_2^{P\to V} \leftarrow \mathsf{TDGen}_2(\mathsf{td}_1^{P\to V})$ along with x_1.
   - **Verifier message:**
     Compute and send $\mathsf{rwi}_2^{V\to P} \leftarrow \mathsf{RWI}_2(\mathsf{wi}_1^{P\to V})$, $\mathsf{c}_2^{V\to P} \leftarrow \mathsf{PZK.ECom}_2(\mathsf{c}_1^{P\to V})$.
3. **Round 3:**
   - **Prover message:**
     - Compute $\mathsf{c}_3^{P\to V} \leftarrow \mathsf{PZK.ECom}_3(\bot, \mathsf{c}_1^{P\to V}, \mathsf{c}_2^{V\to P}; \mathsf{r_c})$.
     - Generate $\mathsf{rwi}_3^{P\to V} \leftarrow \mathsf{RWI}_3(\mathsf{rwi}_1^{P\to V}, \mathsf{rwi}_2^{V\to P}, \mathsf{st_{RWI}}, \mathsf{w_{RWI}})$ for the statement $\mathsf{st_{RWI}} = (x, \mathsf{c}_1^{P\to V}, \mathsf{c}_2^{V\to P}, \mathsf{c}_3^{P\to V}, \mathsf{td}_1^{V\to P}, \mathsf{td}_2^{P\to V}, \mathsf{td}_3^{V\to P})$ using witness $(w, \bot, \bot)$.
     - Send $(x_2, \mathsf{c}_3^{P\to V}, \mathsf{rwi}_3^{P\to V})$.
   - **Verifier message:**
     Compute and send $\mathsf{td}_3^{V\to P} \leftarrow \mathsf{TDGen}_3(\mathsf{td}_1^{V\to P}, \mathsf{td}_2^{P\to V}; \mathsf{r_{td}})$ using randomness $\mathsf{r_{td}}$.
4. **Verifier Output:**
   Output $\mathsf{RWI}_4(\mathsf{rwi}_1^{P\to V}, \mathsf{rwi}_2^{V\to P}, \mathsf{rwi}_3^{P\to V}, \mathsf{st_{RWI}})$.

Valid(Trans):
Given the transcript of the protocol execution, output 1 if $\mathsf{TDOut}(\mathsf{td}_1^{V\to P}, \mathsf{td}_2^{P\to V}, \mathsf{td}_3^{V\to P}) = 1$.

Figure 5: 3 round Promise ZK argument.

adversary to design an adversary $\mathcal{A}_{\mathsf{TDGen}}$ that breaks the "1-rewinding security" of the trapdoor generation protocol TDGen as defined in Section 4 with non-negligible probability.

$\mathcal{A}_{\mathsf{TDGen}}$ interacts with a challenger $\mathcal{C}_{\mathsf{TDGen}}$ and receives a first round message td_1 corresponding to the protocol TDGen. $\mathcal{A}_{\mathsf{TDGen}}$ performs the role of V and interacts with P^*. Now, in round 1 of protocol π, $\mathcal{A}_{\mathsf{TDGen}}$ sets $\mathsf{td}_1^{V\to P}$ as td_1 received from $\mathcal{C}_{\mathsf{TDGen}}$. On receiving a value $\mathsf{td}_2^{P\to V}$ from P^* in round 2, $\mathcal{A}_{\mathsf{TDGen}}$ forwards this message to $\mathcal{C}_{\mathsf{TDGen}}$ as its second round message for the protocol TDGen. $\mathcal{A}_{\mathsf{TDGen}}$ receives td_3 from $\mathcal{C}_{\mathsf{TDGen}}$ which is set as $\mathsf{td}_3^{V\to P}$ in its interaction with P^*. $\mathcal{A}_{\mathsf{TDGen}}$ participates in the rest of the interaction as described in protocol π^{PZK}. Then, $\mathcal{A}_{\mathsf{TDGen}}$ rewinds the adversary P^* back to the beginning of round 2. That is, $\mathcal{A}_{\mathsf{TDGen}}$ creates a look-ahead thread that runs only rounds 2 and 3 of protocol π^{PZK}. As in the main thread, $\mathcal{A}_{\mathsf{TDGen}}$ forwards the adversary's message $\mathsf{td}_2^{P\to V}$ from P^* in round 2 to $\mathcal{C}_{\mathsf{TDGen}}$ and receives td_3 from $\mathcal{C}_{\mathsf{TDGen}}$ which is set as $\mathsf{td}_3^{V\to P}$ in its interaction with P^*.

Now, $\mathcal{A}_{\mathsf{TDGen}}$ runs the extractor $\mathsf{Ext}_{\mathsf{PZK.ECom}}$ of the extractable commitment scheme using the messages in both the threads that correspond to the extractable commitment from P^* to the verifier.

Let the output of $\mathsf{Ext}_{\mathsf{PZK.ECom}}$ be t^*. $\mathcal{A}_{\mathsf{TDGen}}$ outputs t^* as a valid trapdoor to $\mathcal{C}_{\mathsf{TDGen}}$.

Let's analyze why this works. We know that the underlying bounded rewinding secure WI protocol RWI is sound except with negligible probability ϵ. Therefore, since the verifier (in this case $\mathcal{A}_{\mathsf{TDGen}}$) accepts the argument given by P^* with non-negligible probability, from the soundness of RWI and the fact that $x^* \notin L$, it must be the case that P^* used the alternate witness in the RWI arguments produced in both the threads. Therefore, with non-negligible probability $p^2 \cdot (1-\epsilon)$, the adversary P^*, using the non-malleable commitment, commits to a valid trapdoor t^* for the trapdoor generation messages of the verifier TDGen. Let's say the extractor $\mathsf{Ext}_{\mathsf{PZK.ECom}}$ is successful with non-negligible probability q. Therefore, with non-negligible probability $p^2 \cdot q \cdot (1-\epsilon)$, $\mathcal{A}_{\mathsf{TDGen}}$ outputs t^* as a valid trapdoor to $\mathcal{C}_{\mathsf{TDGen}}$ which breaks the 1-rewinding security of the trapdoor generation protocol TDGen which is a contradiction.

5.4.2 Distributional Promise ZK

We now show that the above protocol satisfies the Distributional Promise ZK property against non-adaptive malicious verifiers. In order to do that, we need to construct a simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ satisfying Definition 5 . Lets consider a malicious verifier V^*. Let x denote the external statement sample given as input to Sim without an accompanying witness.

Before we describe the details, lets recall the basic strategy followed by a rewinding simulator. Sim creates a "main thread" of execution that will be actually output at the end of the simulation and a set of "look-ahead" threads that will facilitate the extraction of the adversary's trapdoor. Each look-ahead thread created by Sim shares the first round with the main thread, but contains different messages in the second and third rounds. Sim will use the adversary's messages in the third round of these look-ahead threads to extract the adversary's trapdoor and then use this appropriately in the main thread to simulate the adversary's final view.

Running time of Sim_2: The number of look-ahead threads created is $(\lambda \cdot \frac{1}{p})$ where p is the value that is given as input to Sim_2. Hence, it is easy to see that the running time of Sim_2 is indeed $\mathsf{poly}(\lambda) \cdot O(\frac{1}{p})$.

To prevent a cluttered description, we overload notation when referring to the same object in the main thread and the look-ahead threads. However, it will be clear from context which thread's object is being referred to.

The simulation strategy is described in Figure 6.

5.4.3 Hybrids

We now show that the above simulation strategy is successful. We will show this via a series of computationally indistinguishable hybrids where the first hybrid Hyb_0 corresponds to real world where V^* interacts with an honest prover P and the last hybrid Hyb_4 corresponds to the simulated world where V^* interacts with the above simulator Sim.

First, assume by contradiction that there exists an adversary $\mathcal{A}$ that can distinguish the real and ideal worlds with some probability greater than ϵ where $\epsilon > \lambda^{-c}$ for some constant $c > 0$. Then it must be the case that in the ideal world, $p = \mathsf{pExtract}_c^{V^*}(\mathsf{msg}, \mathsf{st}) \neq 0$. This is because, when $p = 0$, the real and ideal views are identical. Therefore, in the other case, we have $q_{\mathsf{msg}} > p > \epsilon$.

- Hyb_0 **- Real World:** In this hybrid, consider a simulator $\mathsf{Sim}_{\mathsf{Hyb}}$ that plays the role of the honest prover.

$\mathsf{Sim}_1(z)$**:**
Recall that by definition $\mathsf{Sim}_1(z) = P_1(1^\lambda)$. That is, it does the following:

- Compute and send $(\mathsf{rwi}_1^{P\to V}) \leftarrow \mathsf{RWI}_1(1^\lambda)$, $\mathsf{c}_1^{P\to V} \leftarrow \mathsf{PZK.ECom}_1(\mathsf{r_c})$ using a random string $\mathsf{r_c}$. Note that this denotes the output msg with the associated state st being $\mathsf{r_c}$ and the randomness used to generate $\mathsf{rwi}_1^{P\to V}$.
- Receive $\mathsf{td}_1^{V\to P}$ from V^*.

$\mathsf{Sim}_2(z, x, \mathsf{st}, p)$:
Let $p = \mathsf{pExtract}_c^{V^*}(\mathsf{msg}, \mathsf{st})$.
If $p = 0$, sample $(x, w) \leftarrow (\mathcal{X}_\lambda, \mathcal{W}_\lambda)$ and output $\langle P_2(x, w, \mathsf{st}), V^*\rangle$.
Else:

1. **Round 2:**
   - Create a set of $(\lambda \cdot \frac{1}{p})$ look-ahead threads that run only rounds 2 and 3 of the protocol.
   - In each look-ahead thread, sample $(x = (x_1, x_2), w) \leftarrow (\mathcal{X}_\lambda, \mathcal{W}_\lambda)$.
   - Then, in each of the threads (main and look-ahead) :
     - Compute and send $\mathsf{td}_2^{P\to V} \leftarrow \mathsf{TDGen}_2(\mathsf{td}_1^{P\to V})$ along with x_1.
     - Receive $\mathsf{rwi}_2^{V\to P}, \mathsf{c}_2^{V\to P}$
2. **Round 3:**
   In each look-ahead thread:
   - Compute $\mathsf{c}_3^{P\to V} \leftarrow \mathsf{PZK.ECom}_3(\perp, \mathsf{c}_1^{P\to V}, \mathsf{c}_2^{V\to P}; \mathsf{r_c})$.
   - Compute $\mathsf{rwi}_3^{P\to V} \leftarrow \mathsf{RWI}_3(\mathsf{rwi}_1^{P\to V}, \mathsf{rwi}_2^{V\to P}, \mathsf{st}_{\mathsf{RWI}}, \mathsf{w}_{\mathsf{RWI}})$ for the statement $\mathsf{st}_{\mathsf{RWI}} = (x, \mathsf{c}_1^{P\to V}, \mathsf{c}_2^{V\to P}, \mathsf{c}_3^{P\to V}, \mathsf{td}_1^{V\to P}, \mathsf{td}_2^{P\to V}, \mathsf{td}_3^{V\to P})$ using witness $(w, \perp, \perp)$.
   - Send $(x_2, \mathsf{rwi}_3^{P\to V})$.
   - Receive $\mathsf{td}_3^{V\to P}$.

   Input Extraction:
   - Run the trapdoor extractor using the trapdoor generation messages of all the look-ahead threads. That is, compute $\mathsf{t}_V \leftarrow \mathsf{TDExt}(\mathsf{td}_1^{V\to P}, \{\mathsf{td}_2^{P\to V}, \mathsf{td}_3^{V\to P}\})$ where the set denotes the pair of values from all the look-ahead threads.
   - Output "Special Abort" if TDExt fails.

   Main thread:
   - Compute $\mathsf{c}_3^{P\to V} \leftarrow \mathsf{PZK.ECom}_3(\mathsf{t}_V, \mathsf{c}_1^{P\to V}, \mathsf{c}_2^{V\to P}; \mathsf{r_c})$.
   - Compute $\mathsf{rwi}_3^{P\to V} \leftarrow \mathsf{RWI}_3(\mathsf{rwi}_1^{P\to V}, \mathsf{rwi}_2^{V\to P}, \mathsf{st}_{\mathsf{RWI}}, \mathsf{w}_{\mathsf{RWI}})$ for the statement $\mathsf{st}_{\mathsf{RWI}} = (x, \mathsf{c}_1^{P\to V}, \mathsf{c}_2^{V\to P}, \mathsf{c}_3^{P\to V}, \mathsf{td}_1^{V\to P}, \mathsf{td}_2^{P\to V}, \mathsf{td}_3^{V\to P})$ using witness $(\perp, \mathsf{t}_V, \mathsf{r_c})$. Note that the x here denotes the external sample given as input.
   - Send $(x_2, \mathsf{rwi}_3^{P\to V})$.
   - Receive $\mathsf{td}_3^{V\to P}$.

Figure 6: Simulator's description.

- Hyb_1 **- Extraction:** In this hybrid, $\mathsf{Sim}_{\mathsf{Hyb}}$ first runs the protocol honestly $\lambda \cdot \frac{1}{\epsilon}$ times (via rewinding). If the number of executions in which adversary doesn't abort at the end of

the execution is less than λ, then $\mathsf{Sim}_{\mathsf{Hyb}}$ completes the main thread by running the honest prover strategy exactly as in Hyb_0 and stops here. Else, $\mathsf{Sim}_{\mathsf{Hyb}}$ rewinds back to the end of round 1 and creates a fresh set of $(\lambda \cdot \frac{1}{\epsilon})$ look-ahead threads. In all the look-ahead threads, $\mathsf{Sim}_{\mathsf{Hyb}}$ performs just like the honest prover exactly as in Hyb_0. Additionally, $\mathsf{Sim}_{\mathsf{Hyb}}$ also runs the "Input Extraction" phase described in round 3 of the description of Sim to extract the trapdoor t_V. Finally, $\mathsf{Sim}_{\mathsf{Hyb}}$ completes the main thread by running the honest prover strategy exactly as in Hyb_0.

Observe that $\frac{1}{\epsilon} = \lambda^c$ and hence the running time of $\mathsf{Sim}_{\mathsf{Hyb}}$ is $\mathsf{poly}(\lambda)$.

- Hyb_2 **- Changing Commitment:** In the main thread, $\mathsf{Sim}_{\mathsf{Hyb}}$ does the following: in round 3, compute $\mathsf{c}_3^{P\to V} = \mathsf{PZK.ECom}_3(\mathsf{t}_V, \mathsf{c}_1^{P\to V}, \mathsf{c}_2^{V\to P}; \mathsf{r}_{\mathsf{c}}^{P\to V})$.
- Hyb_3 **- Switching WI proofs:** In the main thread, $\mathsf{Sim}_{\mathsf{Hyb}}$ does the following: in round 3, compute $\mathsf{rwi}_3^{P\to V} \leftarrow \mathsf{RWI}_3(\mathsf{rwi}_1^{P\to V}, \mathsf{rwi}_2^{V\to P}, \mathsf{st}_{\mathsf{RWI}}, \mathsf{w}_{\mathsf{RWI}})$ for the statement $\mathsf{st}_{\mathsf{RWI}} = (x, \mathsf{c}_1^{P\to V}, \mathsf{c}_2^{V\to P}, \mathsf{c}_3^{P\to V}, \mathsf{td}_1^{V\to P}, \mathsf{td}_2^{P\to V}, \mathsf{td}_3^{V\to P})$ using witness $(\perp, \mathsf{t}_V, \mathsf{r}_{\mathsf{c}})$.
- Hyb_4 **- Using** $\mathsf{pExtract}_c$**:** In this hybrid, the number of look-ahead threads created is $(\lambda \cdot \frac{1}{p})$. Also, $\mathsf{Sim}_{\mathsf{Hyb}}$ no longer samples (x, w) and instead uses the external input x. The description of $\mathsf{Sim}_{\mathsf{Hyb}}$ in this hybrid matches the description of the simulator Sim. Now, the running time of $\mathsf{Sim}_{\mathsf{Hyb}} = \mathsf{poly}(\lambda) \cdot \frac{1}{p}$.

We now prove that every pair of consecutive hybrids is indistinguishable except with probability at most $\frac{\epsilon}{10}$ and this completes the proof.

Claim 3. *Assuming the existence of the trapdoor extractor* TDExt *for the trapdoor generation protocol* TDGen*,* Hyb_0 *is computationally indistinguishable from* Hyb_1 *except with probability atmost* $\frac{\epsilon}{10}$*.*

Proof. First, let's non-uniformly fix a first round message msg from $\mathsf{Sim}_{\mathsf{Hyb}}$. That is, this is the first round message which maximizes the adversary's probability of success.

Case 1: In Hyb_1, $q_{\mathsf{msg}} < \frac{\epsilon}{2}$.
That is, the probability that the adversary doesn't abort, conditioned on the first message is lesser than $\frac{\epsilon}{2}$. Then in this case, by applying the Chernoff bound, the number of non-aborting executions in the first step of Hyb_1 is lesser than λ except with $2^{-\epsilon}$ probability in which case both hybrids look identical as they just run the honest prover's algorithm and stop at the end of the protocol.

Case 2: suppose $q_{\mathsf{msg}} > 2 \cdot \epsilon$
Then, by the Chernoff bound, except with $2^{-\epsilon}$ probability, in this case, the number of non-aborting transcripts is larger than λ and so $\mathsf{Sim}_{\mathsf{Hyb}}$ proceeds to the next step in Hyb_1. In that case, the only difference between the two hybrids now is that the simulator outputs "Special Abort" in the input extraction phase in Hyb_1. To prove that the two hybrids are indistinguishable, we will now show that the $\mathsf{Pr}[$ $\mathsf{Sim}_{\mathsf{Hyb}}$ outputs "Special Abort" $] \leq \frac{\epsilon}{10}$ in Hyb_1. $\mathsf{Sim}_{\mathsf{Hyb}}$ outputs "Special Abort" only if the algorithm TDExt fails.

By the definition of the scheme TDGen, the algorithm TDExt is successful except with negligible probability if given as input $(\mathsf{td}_1, \{\mathsf{td}_2^i, \mathsf{td}_3^i\}_{i=1}^3)$ such that $\mathsf{TDOut}(\mathsf{td}_1, \mathsf{td}_2^i, \mathsf{td}_3^i) = 1$ for all i. That is, TDExt is successful except with negligible probability if given as input 3 valid executions of the protocol TDGen.

Recall that $\Pr_{(x,w\leftarrow X,W)}[\mathsf{Valid}(\mathsf{msg}, \mathsf{Trans}(P_2(x, w, \mathsf{st}), V^*)) = 1 | (\mathsf{msg}, \cdot) \leftarrow P_1(1^\lambda)] = q_{\mathsf{msg}}$ where the probability is over the random choices of (x, w) and the coins of the parties P_2, V^*. Therefore,

in each look-ahead thread, $Pr[\mathsf{TDOut}(\mathsf{td}_1^{V^*\to P}, \mathsf{td}_2^{P\to V^*}, \mathsf{td}_3^{V^*\to P}) = 1] = q_{\mathsf{msg}}$. Note that this condition holds even in each look-ahead thread because each look-ahead thread only performs an honest execution of the protocol. Recall that $q_{\mathsf{msg}} > \epsilon$. Therefore, in $(\frac{3}{\epsilon})$ expected number of threads, the malicious verifier outputs 3 correct executions of the trapdoor generation protocol. Hence, by using the Markov inequality, in $(\lambda \cdot \frac{1}{\epsilon})$ threads, the extraction is successful except with negligible probability and this completes the proof.

Case 3: suppose $2 \cdot \epsilon > q_{\mathsf{msg}} > \frac{\epsilon}{2}$

Now, suppose the number of non-aborting transcripts was lesser than λ. Then, the two hybrids are indistinguishable as in case 1. Similarly, if the number of non-aborting transcripts was greater than λ, the two hybrids are indistinguishable as in case 2. □

Claim 4. *Assuming the hiding property of the extractable commitment scheme* $\mathsf{PZK.ECom}$*,* Hyb_1 *is computationally indistinguishable from* Hyb_2 *except with probability at most* $\frac{\epsilon}{10}$*.*

Proof. The only difference between Hyb_1 and Hyb_2 is that in Hyb_2, the simulator now computes the extractable commitment using the adversary's trapdoor value. Suppose there exists an adversary V^* that can distinguish between the two hybrids with non-negligible probability greater than $\frac{\epsilon}{10}$. We will use V^* to design an adversary $\mathcal{A}_{\mathsf{Hid}}$ that breaks the hiding of the extractable commitment scheme.

$\mathcal{A}_{\mathsf{Hid}}$ interacts with a challenger $\mathcal{C}_{\mathsf{Hid}}$. $\mathcal{A}_{\mathsf{Hid}}$ performs the role of $\mathsf{Sim}_{\mathsf{Hyb}}$ in its interaction with V^* almost exactly as done in Hyb_1. $\mathcal{A}_{\mathsf{Hid}}$ interacts with a challenger $\mathcal{C}_{\mathsf{Hid}}$ and receives a first round message which is set as $\mathsf{c}_1^{P\to V}$ in its interaction with V^* in round 1 of protocol π^{PZK}. $\mathcal{A}_{\mathsf{Hid}}$ receives a value $\mathsf{c}_2^{V\to P}$ in round 2 of protocol π^{PZK} on the main thread, which it sends to $\mathcal{C}_{\mathsf{Hid}}$ as its second round message. Then, in each look-ahead thread, $\mathcal{A}_{\mathsf{Hid}}$ performs round 3 exactly as in Hyb_1. In particular, $\mathcal{A}_{\mathsf{Hid}}$ sends a random string as its message $\mathsf{c}_3^{P\to V}$. From the properties of the scheme $\mathsf{PZK.ECom}$, recall that $\mathsf{PZK.ECom}_3(\perp)$ is indistinguishable from a random string and so $\mathcal{A}_{\mathsf{Hid}}$ can generate this by picking a uniformly random string without knowing the randomness used to generate $\mathsf{c}_1^{P\to V}$.

Then, after extracting the trapdoor t_V in the input extraction phase, $\mathcal{A}_{\mathsf{Hid}}$ sends the pair of values $(\perp, \mathsf{t}_V)$ to $\mathcal{C}_{\mathsf{Hid}}$. Recall that $\mathsf{PZK.ECom}$ is a delayed-input scheme. $\mathcal{A}_{\mathsf{Hid}}$ receives a third round message from $\mathcal{C}_{\mathsf{Hid}}$ which is either a commitment to $\perp$ or t_V. This is sent to V^* as the value $\mathsf{c}_3^{P\to V}$ in the main thread.

Observe that the first case corresponds to Hyb_1 while the second case corresponds to Hyb_2. Therefore, if the adversary V^* can distinguish between these two hybrids with non-negligible probability greater than $\frac{\epsilon}{10}$, $\mathcal{A}_{\mathsf{Hid}}$ will use the same guess to break the hiding of the extractable commitment scheme $\mathsf{PZK.ECom}$ with non-negligible probability which is a contradiction. □

Claim 5. *Assuming* RWI *is a witness indistinguishable argument system with bounded-rewinding security,* Hyb_2 *is computationally indistinguishable from* Hyb_3 *except with probability atmost* $\frac{\epsilon}{10}$*.*

Proof. The only difference between Hyb_2 and Hyb_3 is that in Hyb_3, in the main thread, the simulator now computes the bounded rewinding secure WI proof using a witness for the alternate statement. Suppose there exists an adversary V^* that can distinguish between the two hybrids with non-negligible probability greater than $\frac{\epsilon}{10}$. We will use V^* to design an adversary $\mathcal{A}_{\mathsf{RWI}}$ that breaks the security of the bounded rewinding secure WI scheme.

$\mathcal{A}_{\mathsf{RWI}}$ interacts with a challenger $\mathcal{C}_{\mathsf{RWI}}$. $\mathcal{A}_{\mathsf{RWI}}$ performs the role of $\mathsf{Sim}_{\mathsf{Hyb}}$ in its interaction with V^* almost exactly as done in Hyb_2. From $\mathcal{C}_{\mathsf{RWI}}$, $\mathcal{A}_{\mathsf{RWI}}$ receives a first round message rwi_1 which is set as $\mathsf{rwi}_1^{P\to V}$ in its interaction with V^* in round 1 of protocol π^{PZK}. Then, $\mathcal{A}_{\mathsf{RWI}}$ creates a set of L

look ahead threads (L being the parameter for the protocol RWI which is set to 5 here). For each thread, on receiving $\mathsf{rwi}_2^{V\to P}$ in round 2, $\mathcal{A}_{\mathsf{RWI}}$ forwards this to $\mathcal{C}_{\mathsf{RWI}}$ as its second round message.

For each thread, $\mathcal{A}_{\mathsf{RWI}}$ sends the statement $\mathsf{st}_{\mathsf{RWI}} = (x, \mathsf{c}_1^{P\to V}, \mathsf{c}_2^{V\to P}, \mathsf{c}_3^{P\to V}, \mathsf{td}_1^{V\to P}, \mathsf{td}_2^{P\to V}, \mathsf{td}_3^{V\to P})$ to $\mathcal{C}_{\mathsf{RWI}}$ where the other values are generated as in Hyb_2. In each look-ahead thread, $\mathcal{A}_{\mathsf{RWI}}$ also sends the witness $(w, \perp, \perp)$ to $\mathcal{C}_{\mathsf{RWI}}$. In the main thread, $\mathcal{A}_{\mathsf{RWI}}$ sends the pair of witnesses $(w, \perp, \perp)$ and $(\perp, \mathsf{t}_V, \mathsf{r_c})$ where t_V is generated in the input extraction phase. Recall that RWI is a delayed-input scheme. For each thread, $\mathcal{A}_{\mathsf{RWI}}$ receives a third round message rwi_3 which is set as $\mathsf{rwi}_3^{P\to V}$ in its interaction with V^* in round 3 of protocol π^{PZK}. The rest of protocol π^{PZK} is performed exactly as in Hyb_2. Observe that on each thread, since V^* produces a non-aborting transcript with non-negligible probability, $\mathcal{A}_{\mathsf{RWI}}$ (via the trapdoor extractor TDExt) extracts a valid trapdoor with non-negligible probability.

Observe that the first case corresponds to Hyb_2 while the second case corresponds to Hyb_3. Therefore, if the adversary V^* can distinguish between these two hybrids with non-negligible probability greater than $\frac{\epsilon}{10}$, $\mathcal{A}_{\mathsf{RWI}}$ will use the same guess to break the rewinding security of the scheme RWI with non-negligible probability which is a contradiction. □

Claim 6. *Assuming the existence of the trapdoor extractor* TDExt *for the trapdoor generation protocol* TDGen*,* Hyb_3 *is computationally indistinguishable from* Hyb_4 *except with probability at most* $\frac{\epsilon}{10}$*.*

Proof. First, let's non-uniformly fix a first round message msg from $\mathsf{Sim}_{\mathsf{Hyb}}$. That is, this is the first round message which maximizes the adversary's probability of success.

Case 1: suppose $q_{\mathsf{msg}} < \frac{\epsilon}{2}$
Then, clearly the output of $\mathsf{pExtract}$ is 0 in Hyb_4. That is, the probability that the adversary doesn't abort, conditioned on the first message is lesser than ϵ. Then clearly, in this case, except with probability $\leq \epsilon$, both hybrids look identical as they just run the honest prover's algorithm.

Case 2: suppose $q_{\mathsf{msg}} > 2 \cdot \epsilon$
Then, in this case, the output of $\mathsf{pExtract}$ is not 0. Also, except with probability lesser than ϵ, in this case, the number of non-aborting transcripts is larger than λ in Hyb_3 and so $\mathsf{Sim}_{\mathsf{Hyb}}$ proceeds to the next step in Hyb_3.

Observe that the only difference between the two hybrids is that the simulator might output "Special Abort" in the input extraction phase of Hyb_4 which happens in Hyb_3 only with negligible probability. To prove that the two hybrids are indistinguishable, we will now show that the $\mathsf{Pr}[$ $\mathsf{Sim}_{\mathsf{Hyb}}$ outputs "Special Abort" $] \leq \frac{\epsilon}{10}$ in Hyb_4. $\mathsf{Sim}_{\mathsf{Hyb}}$ outputs "Special Abort" only if the algorithm TDExt fails.

By the definition of the scheme TDGen, the algorithm TDExt is successful except with negligible probability if given as input $(\mathsf{td}_1, \{\mathsf{td}_2^i, \mathsf{td}_3^i\}_{i=1}^3)$ such that $\mathsf{TDOut}(\mathsf{td}_1, \mathsf{td}_2^i, \mathsf{td}_3^i) = 1$ for all i. That is, TDExt is successful except with negligible probability if given as input 3 valid executions of the protocol TDGen.

Recall that $\Pr_{(x,w\leftarrow X,W)}[\mathsf{Valid}(\mathsf{msg}, \mathsf{Trans}(P_2(x,w,\mathsf{st}), V^*)) = 1 | (\mathsf{msg}, \cdot) \leftarrow P_1(1^\lambda)] = q_{\mathsf{msg}}$ where the probability is over the random choices of (x, w) and the coins of the parties P_2, V^*. Therefore, in each look-ahead thread, $Pr[\mathsf{TDOut}(\mathsf{td}_1^{V\to P}, \mathsf{td}_2^{P\to V}, \mathsf{td}_3^{V\to P}) = 1] = q_{\mathsf{msg}}$. Note that this condition holds even in each look-ahead thread because each look-ahead thread only performs an honest execution of the protocol. Recall that $q_{\mathsf{msg}} > p$. Therefore, in $(\frac{3}{p})$ expected number of threads, the malicious verifier outputs 3 correct executions of the trapdoor generation protocol. Hence, by using the Markov inequality, in $(\lambda \cdot \frac{1}{p})$ threads, the extraction is successful except with negligible probability and this completes the proof.

Case 3: suppose $2 \cdot \epsilon > q_{\mathsf{msg}} > \epsilon$
This is similar to the proof of case 3 in Claim 3. That is, suppose $\mathsf{pExtract}$ outputs 0, then the two hybrids output identical threads using the honest prover's algorithm and hence are indistinguishable. On the other hand, if $\mathsf{pExtract}$ output some value $p > \epsilon$, the extraction would succeed except with negligible probability and hence the two hybrids would be indistinguishable.

□

6 Four Round Malicious Secure MPC

Let f be any functionality. Consider n parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ with inputs $\mathsf{x}_1, \ldots, \mathsf{x}_n$ respectively who wish to compute f on their joint inputs by running a secure multiparty computation (MPC) protocol. Let π^{SM} be any 3 round protocol that runs without any setup for the above task and is secure against adversaries that can be completely malicious in the first round, semi-malicious in the next two rounds and can corrupt upto $(n-1)$ parties. In this section, we show how to generically transform π^{SM} into a 4 round protocol π without setup and secure against malicious adversaries that can corrupt upto $(n-1)$ parties. Formally, we prove the following theorem:

Theorem 7. *Based on the existence of polynomially secure:*

- *DDH/Quadratic Residuosity/N^{th} Residuosity assumption AND*
- *a 3 round MPC protocol for any functionality f that is secure against malicious adversaries in the first round and semi-malicious adversaries in the next two rounds,*

the protocol π presented below is a 4 round MPC protocol for any functionality f, in the plain model.

We can instantiate the underlying MPC protocol with the construction of Brakerski et al.[BHP17], which satisfies our requirements. That is:

Imported Lemma 1. *([BHP17]): There exists a 3 round MPC protocol for any functionality f based on the LWE assumption that is secure against malicious adversaries in the first round and semi-malicious adversaries in the next 2 rounds.*

Formally, we obtain the following corollary on instantiating the MPC protocol:

Corollary 8. *Assuming the existence of polynomially secure :*

- *DDH/Quadratic Residuosity/N^{th} Residuosity assumption AND*
- *LWE*

the protocol π presented below is a 4 round MPC protocol for any functionality f, in the plain model.

We first list some notation and the primitives used before describing the construction.

Primitives used:

- Let NCom be any non-interactive commitment scheme. We know that such a commitment scheme can be built assuming injective one way functions.

- $\mathsf{WZK} = (\mathsf{WZK}_1, \mathsf{WZK}_2, \mathsf{WZK}_3, \mathsf{WZK}_4)$ is a three-message weak zero knowledge argument with a delayed-input property: that is, the first round algorithm of the prover WZK_1 does not take the statement or witness as input. Jain et al.[JKKR17] constructed such a scheme based on DDH/Quadratic Residuosity/N^{th} Residuosity assumption.

- $\mathsf{R.ECom} = (\mathsf{R.ECom}_1, \mathsf{R.ECom}_2, \mathsf{R.ECom}_3)$ is the three-message extractable commitment scheme defined in Section 4.4. The protocol has a delayed-input property: that is, the first round algorithm of the committer $\mathsf{R.ECom}_1$ does not take the message to be committed as input. Also, recall that the construction of $\mathsf{R.ECom}$ internally uses a standard non-interactive commitment scheme - Com, a pseudorandom function PRF and an underlying extractable commitment scheme $\mathsf{BRew.ECom}$. In the construction, we use a parameter K which we set to be 4 here.

  Recall that we also define a property called "well-formedness" of the commitment scheme. $\mathsf{R.ECom}$ has an associated PPT algorithm $\mathsf{Ext}_{\mathsf{R.ECom}}$ that, given 5 "well-formed" executions of the $\mathsf{R.ECom}$ protocol, outputs the committed value with overwhelming probability.

- $\mathsf{TDGen} = (\mathsf{TDGen}_1, \mathsf{TDGen}_2, \mathsf{TDGen}_3, \mathsf{TDOut})$ is the three-message trapdoor generation protocol as defined in Section 4. The first 3 algorithms are used to generate the messages of the protocol while TDOut checks that the execution was honest. TDGen also has two associated PPT algorithms $\mathsf{TDValid}, \mathsf{TDExt}$.
  Recall that the algorithm $\mathsf{TDValid}$ takes as input a tuple of 4 values : the first three being outputs of the algorithms $\mathsf{TDGen}_1, \mathsf{TDGen}_2, \mathsf{TDGen}_3$ and outputs 1 if the fourth value is a valid trapdoor with respect to these three.
  The algorithm TDExt, as defined earlier, with overwhelming probability, outputs a valid trapdoor if given honestly generated messages of the trapdoor generator in 3 executions with a common first round.

- $\mathsf{WI} = (\mathsf{WI}_1, \mathsf{WI}_2, \mathsf{WI}_3, \mathsf{WI}_4)$ is a three-message witness indistinguishable argument with a delayed-input property: that is, the first round algorithm of the prover WI_1 does not take the statement or witness as input.

- $\mathsf{RWI} = (\mathsf{RWI}_1, \mathsf{RWI}_2, \mathsf{RWI}_3, \mathsf{RWI}_4)$ is the three round delayed-input witness indistinguishable argument with bounded rewinding security defined in Section 4. The fourth algorithm RWI_4 is the final verification algorithm. We will set the rewinding security parameter L to be 5 in all our applications.

- $\mathsf{NMCom} = (\mathsf{NMCom}_1, \mathsf{NMCom}_2, \mathsf{NMCom}_3)$ is the three-message non-malleable commitment scheme with respect to commitment defined in Goyal et al.[GPR16] that can be based on injective one way functions. Recall that the protocol has a delayed-input property. Also, note that in their protocol, the third round message is indistinguishable from a random string when the input message is $\perp$.

  Let $\mathsf{Ext}_{\mathsf{NMCom}}$ denote the polynomial time extractor of this non-malleable commitment scheme. Recall that in the scheme from [GPR16], given as input honestly generated messages from the sender in 2 executions with a common first round, $\mathsf{Ext}_{\mathsf{NMCom}}$ can extract the value committed to inside the non-malleable commitment with noticeable probability.

  The NMCom we use is tagged. In the authenticated channels setting, the tag of each user performing a non-malleable commitment can just be its identity. In the general setting, in the first round, each party can choose a strong digital signature verification key VK and

signing key, and then sign all its messages using this signature scheme for every message sent in the protocol. This VK is then used as the tag for all non-malleable commitments. This ensures that every adversarial party must choose a tag that is different from any tags chosen by honest parties, otherwise the adversary will not be able to sign any of its messages by the existential unforgeability property of the signature scheme. This is precisely the property that is assumed when applying NMCom. For ease of notation, we suppress writing the tags explicitly in our protocols below.

- π^{SM} is a three-round MPC protocol that is secure against malicious adversaries in the first round and semi-malicious adversaries in the next two rounds. Let $(\pi_1^{\mathsf{SM}}, \pi_2^{\mathsf{SM}}, \pi_3^{\mathsf{SM}})$ denote the algorithms used by any party to compute the messages in each of the three rounds and OUT denotes the algorithm to compute the final output. Also, let Trans_i denote all the messages sent in an execution of π^{SM} up to round i. Protocol π^{SM} runs over a broadcast channel. Let $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ denote the straight line simulator for this protocol - that is, $\mathcal{S}_i$ is the simulator's algorithm to compute the i^{th} round messages. Also, the protocol has the following properties that are satisfied by the instantiations:

  1. $\mathcal{S}_1$ and $\mathcal{S}_2$ run the honest party's algorithms but using input 0 - in particular, they don't need the input, randomness and output of the malicious parties.
  2. The algorithm π_1^{SM} doesn't depend on the input. Further, any party can generate round 2 messages by running the algorithm π_2^{SM} even without running the first algorithm π_1^{SM} itself.
  3. The algorithm π_3^{SM} doesn't require any new input or randomness that was not already used in the algorithms $\pi_1^{\mathsf{SM}}, \pi_2^{\mathsf{SM}}$. Looking ahead, this is used in our security proof when we want to invoke the simulator of this protocol π^{SM}, we need to be sure that we have fed the correct input and randomness to the simulator. This is true for the instantiation we consider, where the semi-malicious simulator requires only the secret keys of corrupted parties (that are fixed in the second round) apart from the protocol transcript.

Remark: As mentioned in the introduction, we use the Promise ZK argument system in a non-black box manner in the construction of our 4 round MPC protocol. Additionally, instead of an extractable commitment as a building block in the Promise ZK construction, here, we use a non-malleable commitment that has the same extraction properties.

In our construction, we use proofs for some NP languages that we elaborate on below.

NP language L_1 is characterized by the following relation R_1.
Statement : $\mathsf{st} = (\mathsf{nc})$
Witness : $\mathsf{w} = (\mathsf{r}_{\mathsf{nc}})$
$R(\mathsf{st}, \mathsf{w}) = 1$ if and only if :

- $\mathsf{nc} = \mathsf{Com}(1; \mathsf{r}_{\mathsf{nc}})$

In our protocol, when we use this language for proofs between parties P_i and P_j, where P_i is the prover and P_j is the verifier. For better clarity, we denote this language by $L_1^{i \to j}$.
NP language L_2 is characterized by the following relation R_2.
Statement : $\mathsf{st} = (\mathsf{r.ec}_{a,1}, \mathsf{r.ec}_{a,2}, \mathsf{r.ec}_{a,3}, \mathsf{r.ec}_{b,1}, \mathsf{r.ec}_{b,2}, \mathsf{r.ec}_{b,3}, \mathsf{msg}_2, \mathsf{Trans}, \mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3, \mathsf{td}_1, \mathsf{nc})$
Witness : $\mathsf{w} = (\mathsf{inp}, \mathsf{r}, \mathsf{r}_{a,\mathsf{r.ec}}, \mathsf{r}_{b,\mathsf{r.ec}}, \mathsf{t}, \mathsf{r}_{\mathsf{c}}, \mathsf{r}_{\mathsf{nc}})$
$R(\mathsf{st}, \mathsf{w}) = 1$ if and only if :

1. Either the messages $(\mathsf{r.ec}_{a,1}, \mathsf{r.ec}_{a,2}, \mathsf{r.ec}_{a,3})$ or $(\mathsf{r.ec}_{b,1}, \mathsf{r.ec}_{b,2}, \mathsf{r.ec}_{b,3})$ form a well-formed extractable commitment of $(\mathsf{inp}, \mathsf{r})$ and msg_2 is the second round message generated by running protocol π^{SM} using input inp, randomness r where the protocol transcript at the end of round 1 is Trans (OR)
2. $(\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3)$ form a non-malleable commitment to a value t that is a valid trapdoor for the messages td_1 generated by the trapdoor generator. (OR)
3. nc is a commitment to 0.

Formally, $R(\mathsf{st}, \mathsf{w}) = 1$ if and only if :

- $\mathsf{r.ec}_{a,1} = \mathsf{R.ECom}_1(\mathsf{r}_{a,\mathsf{r.ec}})$ AND
- $\mathsf{r.ec}_{a,3} = \mathsf{R.ECom}_3(\mathsf{inp}, \mathsf{r}, \mathsf{r.ec}_{a,1}, \mathsf{r.ec}_{a,2}; \mathsf{r}_{a,\mathsf{r.ec}})$ AND
- $\mathsf{msg}_2 = \pi_2^{\mathsf{SM}}(\mathsf{inp}, \mathsf{Trans}; \mathsf{r})$ AND
- $(\mathsf{r.ec}_{a,1}, \mathsf{r.ec}_{a,2}, \mathsf{r.ec}_{a,3})$ is "well-formed" as defined in Section 4.3.

(OR)

- $\mathsf{r.ec}_{b,1} = \mathsf{R.ECom}_1(\mathsf{r}_{b,\mathsf{r.ec}})$ AND
- $\mathsf{r.ec}_{b,3} = \mathsf{R.ECom}_3(\mathsf{inp}, \mathsf{r}, \mathsf{r.ec}_{b,1}, \mathsf{r.ec}_{b,2}; \mathsf{r}_{b,\mathsf{r.ec}})$ AND
- $\mathsf{msg}_2 = \pi_2^{\mathsf{SM}}(\mathsf{inp}, \mathsf{Trans}; \mathsf{r})$ AND
- $(\mathsf{r.ec}_{b,1}, \mathsf{r.ec}_{b,2}, \mathsf{r.ec}_{b,3})$ is "well-formed" as defined in Section 4.3.

(OR)

- $\mathsf{TDValid}(\mathsf{td}_1, \mathsf{t}) = 1$ AND
- $\mathsf{c}_1 = \mathsf{NMCom}_1(\mathsf{r}_\mathsf{c})$ AND
- $\mathsf{c}_3 = \mathsf{NMCom}_3(\mathsf{t}, \mathsf{c}_1, \mathsf{c}_2; \mathsf{r}_\mathsf{c})$.

(OR)

- $\mathsf{nc} = \mathsf{Com}(0; \mathsf{r}_{\mathsf{nc}})$.

In our protocol, when we use this language for proofs between parties P_i and P_j, where P_i is the prover and P_j is the verifier. For the trapdoor generation messages in the statement, P_j is the trapdoor generator and P_i is the receiver. For better clarity, we denote this language by $L_2^{i \to j}$.
NP language L_3 is characterized by the following relation R_3.
Statement : $\mathsf{st} = (\mathsf{r.ec}_{a,1}, \mathsf{r.ec}_{a,2}, \mathsf{r.ec}_{a,3}, \mathsf{r.ec}_{b,1}, \mathsf{r.ec}_{b,2}, \mathsf{r.ec}_{b,3}, \mathsf{msg}_3, \mathsf{Trans}, \mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3, \mathsf{td}_1)$
Witness : $\mathsf{w} = (\mathsf{inp}, \mathsf{r}, \mathsf{r}_{a,\mathsf{r.ec}}, \mathsf{r}_{b,\mathsf{r.ec}}, \mathsf{t}, \mathsf{r}_\mathsf{c})$
$R(\mathsf{st}, \mathsf{w}) = 1$ if and only if :

1. Either the messages $(\mathsf{r.ec}_{a,1}, \mathsf{r.ec}_{a,2}, \mathsf{r.ec}_{a,3})$ or $(\mathsf{r.ec}_{b,1}, \mathsf{r.ec}_{b,2}, \mathsf{r.ec}_{b,3})$ form a well-formed extractable commitment of $(\mathsf{inp}, \mathsf{r})$ and msg_2 is the second round message generated by running protocol π^{SM} using input inp, randomness r where the protocol transcript at the end of round 1 is Trans (OR)

2. $(\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3)$ form a non-malleable commitment to a value t that is a valid trapdoor for the message td_1 generated by the trapdoor generator.

Formally, $R(\mathsf{st}, \mathsf{w}) = 1$ if and only if :

- $\mathsf{r.ec}_{a,1} = \mathsf{R.ECom}_1(\mathsf{r}_{a,\mathsf{r.ec}})$ AND
- $\mathsf{r.ec}_{a,3} = \mathsf{R.ECom}_3(\mathsf{inp}, \mathsf{r}, \mathsf{r.ec}_{a,1}, \mathsf{r.ec}_{a,2}; \mathsf{r}_{a,\mathsf{r.ec}})$ AND
- $\mathsf{msg}_2 = \pi_2^{\mathsf{SM}}(\mathsf{inp}, \mathsf{Trans}; \mathsf{r})$ AND
- $(\mathsf{r.ec}_{a,1}, \mathsf{r.ec}_{a,2}, \mathsf{r.ec}_{a,3})$ is "well-formed" as defined in Section 4.3.

(OR)

- $\mathsf{r.ec}_{b,1} = \mathsf{R.ECom}_1(\mathsf{r}_{b,\mathsf{r.ec}})$ AND
- $\mathsf{r.ec}_{b,3} = \mathsf{R.ECom}_3(\mathsf{inp}, \mathsf{r}, \mathsf{r.ec}_{b,1}, \mathsf{r.ec}_{b,2}; \mathsf{r}_{b,\mathsf{r.ec}})$ AND
- $\mathsf{msg}_2 = \pi_2^{\mathsf{SM}}(\mathsf{inp}, \mathsf{Trans}; \mathsf{r})$ AND
- $(\mathsf{r.ec}_{b,1}, \mathsf{r.ec}_{b,2}, \mathsf{r.ec}_{b,3})$ is "well-formed" as defined in Section 4.3.

(OR)

- $\mathsf{TDValid}(\mathsf{td}_1, \mathsf{t}) = 1$ AND
- $\mathsf{c}_1 = \mathsf{NMCom}_1(\mathsf{r}_\mathsf{c})$ AND
- $\mathsf{c}_3 = \mathsf{NMCom}_3(\mathsf{t}, \mathsf{c}_1, \mathsf{c}_2; \mathsf{r}_\mathsf{c})$.

In our protocol, when we use this language for proofs between parties P_i and P_j, where P_i is the prover and P_j is the verifier. For the trapdoor generation messages in the statement, P_j is the trapdoor generator and P_i is the receiver. For better clarity, we denote this language by $L_3^{i \to j}$.

Notation :

- We assume broadcast channels.
- λ denotes the security parameter.
- In the superscript, we use $i \to j$ to denote that the message was sent by party P_i with intended recipient as party P_j. (recall that all messages are broadcast).
- The round number of any sub-protocol being used (such as the non-malleable commitment, bounded rewinding secure WI arguments etc.) is written in the subscript.
- We use two instantiations of the extractable commitment scheme $\mathsf{R.ECom}$; we use $(a, i), (b, i)$ in the subscript to denote each of the instantiations (where i is the round number).

6.1 Protocol

The description of protocol π is as follows:

Inputs: Each party P_i has input x_i and uses randomness r_i to compute the message in each round of the protocol π^{SM}. We now describe the messages sent by party P_i.

1. **Round 1:**
   P_i does the following:
   - Compute $\mathsf{msg}_{1,i} \leftarrow \pi_1^{\mathsf{SM}}(\mathsf{x}_i, ; \mathsf{r}_i)$.
   - For each $j \in [n]$ with $j \neq i$, P_i does the following:
     - Compute $\mathsf{c}_1^{i\to j} \leftarrow \mathsf{NMCom}_1(\mathsf{r}_{\mathsf{c}}^{i\to j}), \mathsf{r.ec}_{a,1}^{i\to j} \leftarrow \mathsf{R.ECom}_1(\mathsf{r}_{a,\mathsf{r.ec}}^{i\to j}), \mathsf{r.ec}_{b,1}^{i\to j} \leftarrow \mathsf{R.ECom}_1(\mathsf{r}_{b,\mathsf{r.ec}}^{i\to j})$ using random strings $\mathsf{r}_{\mathsf{c}}^{i\to j}, \mathsf{r}_{a,\mathsf{r.ec}}^{i\to j}$ and $\mathsf{r}_{b,\mathsf{r.ec}}^{i\to j}$ respectively.
     - Compute $\mathsf{td}_1^{i\to j} \leftarrow \mathsf{TDGen}_1(\mathsf{r}_{\mathsf{td}}^{i\to j})$ using a random string $\mathsf{r}_{\mathsf{td}}^{i\to j}$.
     - Generate $(\mathsf{wi}_1^{i\to j}) \leftarrow \mathsf{WI}_1(1^\lambda)$, $(\mathsf{rwi}_1^{i\to j}) \leftarrow \mathsf{RWI}_1(1^\lambda)$ and $(\mathsf{wzk}_1^{i\to j}) \leftarrow \mathsf{WZK}_1(1^\lambda)$.
   - Broadcast $(\mathsf{msg}_{1,i}, \mathsf{c}_1^{i\to j}, \mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{td}_1^{i\to j}, \mathsf{wi}_1^{i\to j}, \mathsf{rwi}_1^{i\to j}, \mathsf{wzk}_1^{i\to j})$.

2. **Round 2:**
   For each $j \in [n]$ with $j \neq i$, P_i does the following:
   - Compute $\mathsf{c}_2^{i\to j} \leftarrow \mathsf{NMCom}_2(\mathsf{c}_1^{j\to i}), \mathsf{r.ec}_{a,2}^{i\to j} \leftarrow \mathsf{R.ECom}_2(\mathsf{r.ec}_{a,1}^{j\to i}), \mathsf{r.ec}_{b,2}^{i\to j} \leftarrow \mathsf{R.ECom}_2(\mathsf{r.ec}_{b,1}^{j\to i})$ and $\mathsf{td}_2^{i\to j} \leftarrow \mathsf{TDGen}_2(\mathsf{td}_1^{j\to i})$.
   - Compute $\mathsf{wi}_2^{i\to j} \leftarrow \mathsf{WI}_2(\mathsf{wi}_1^{j\to i}), \mathsf{rwi}_2^{i\to j} \leftarrow \mathsf{RWI}_2(\mathsf{rwi}_1^{j\to i})$ and $\mathsf{wzk}_2^{i\to j} \leftarrow \mathsf{WZK}_2(\mathsf{wzk}_1^{j\to i})$.
   - Broadcast $(\mathsf{c}_2^{i\to j}, \mathsf{r.ec}_{a,2}^{i\to j}, \mathsf{r.ec}_{b,2}^{i\to j}, \mathsf{td}_2^{i\to j}, \mathsf{wi}_2^{i\to j}, \mathsf{rwi}_2^{i\to j}, \mathsf{wzk}_2^{i\to j})$ for all $j \in [n]$, $j \neq i$.

3. **Round 3:**
   Let Trans_1 denote the transcript after round 1 of protocol π^{SM}. P_i does the following:
   - Compute $\mathsf{msg}_{2,i} \leftarrow \pi_2^{\mathsf{SM}}(\mathsf{x}_i, \mathsf{Trans}_1; \mathsf{r}_i)$.
   - Compute $\mathsf{nc}_i \leftarrow \mathsf{NCom}(1; \mathsf{r}_{\mathsf{nc},i})$.
   - For each $j \in [n]$ with $j \neq i$, compute:
     - $\mathsf{c}_3^{i\to j} \leftarrow \mathsf{NMCom}_3(\perp, \mathsf{c}_1^{i\to j}, \mathsf{c}_2^{j\to i}; \mathsf{r}_{\mathsf{c}}^{i\to j})$, $\mathsf{r.ec}_{a,3}^{i\to j} \leftarrow \mathsf{R.ECom}_3(\mathsf{x}_i, \mathsf{r}_i, \mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{a,2}^{j\to i}; \mathsf{r}_{a,\mathsf{r.ec}}^{i\to j})$, $\mathsf{r.ec}_{b,3}^{i\to j} \leftarrow \mathsf{R.ECom}_3(\perp, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}; \mathsf{r}_{b,\mathsf{r.ec}}^{i\to j})$ using random strings $\mathsf{r}_{\mathsf{c}}^{i\to j}$, $\mathsf{r}_{a,\mathsf{r.ec}}^{i\to j}$ and $\mathsf{r}_{b,\mathsf{r.ec}}^{i\to j}$.
     - $\mathsf{td}_3^{i\to j} \leftarrow \mathsf{TDGen}_3(\mathsf{td}_1^{i\to j}, \mathsf{td}_2^{j\to i}; \mathsf{r}_{\mathsf{td}}^{i\to j})$ using randomness $\mathsf{r}_{\mathsf{td}}^{i\to j}$.
     - $\mathsf{wzk}_3^{i\to j} \leftarrow \mathsf{WZK}_3(\mathsf{wzk}_1^{i\to j}, \mathsf{wzk}_2^{j\to i}, \mathsf{st}_1^{i\to j}, \mathsf{w}_1^{i\to j})$ for the statement $\mathsf{st}_1^{i\to j} = (\mathsf{nc}_1^{i\to j}) \in L_1^{i\to j}$ using witness $\mathsf{w}_1^{i\to j} = (\mathsf{r}_{\mathsf{nc},i})$.
     - $\mathsf{rwi}_3^{i\to j} \leftarrow \mathsf{RWI}_3(\mathsf{rwi}_1^{i\to j}, \mathsf{rwi}_2^{j\to i}, \mathsf{st}_2^{i\to j}, \mathsf{w}_2^{i\to j})$ for the statement $\mathsf{st}_2^{i\to j} = (\mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{a,2}^{j\to i}, \mathsf{r.ec}_{a,3}^{i\to j}, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}, \mathsf{r.ec}_{b,3}^{i\to j}, \mathsf{msg}_{2,i}, \mathsf{Trans}_1, \mathsf{c}_1^{i\to j}, \mathsf{c}_2^{j\to i}, \mathsf{c}_3^{i\to j}, \mathsf{td}_1^{j\to i}, \mathsf{nc}_i) \in L_2^{i\to j}$ using witness $\mathsf{w}_2^{i\to j} = (\mathsf{x}_i, \mathsf{r}_i, \mathsf{r}_{a,\mathsf{r.ec}}^{i\to j}, \perp, \perp, \perp, \perp)$.
   - Broadcast $(\mathsf{msg}_{2,i}, \mathsf{nc}_i, \mathsf{c}_3^{i\to j}, \mathsf{r.ec}_{a,3}^{i\to j}, \mathsf{r.ec}_{b,3}^{i\to j}, \mathsf{td}_3^{i\to j}, \mathsf{rwi}_3^{i\to j})$ for all $j \in [n]$, $j \neq i$.

4. **Round 4:**
   Let Trans_2 denote the transcript after round 2 of protocol π^{SM}. P_i does the following:
   - For each $j \in [n]$ with $j \neq i$, broadcast an abort signal to all the parties if

- $\mathsf{TDOut}(\mathsf{td}_1^{j\to i}, \mathsf{td}_2^{i\to j}, \mathsf{td}_3^{j\to i}) \neq 1$ (OR)
- $\mathsf{WZK}_4(\mathsf{wzk}_1^{j\to i}, \mathsf{wzk}_2^{i\to j}, \mathsf{wzk}_3^{j\to i}, \mathsf{st}_1^{j\to i}) \neq 1$ where $\mathsf{st}_1^{j\to i} = (\mathsf{nc}_1^{j\to i})$. (OR)
- $\mathsf{RWI}_4(\mathsf{rwi}_1^{i\to j}, \mathsf{rwi}_2^{i\to j}, \mathsf{rwi}_3^{j\to i}, \mathsf{st}_2^{j\to i}) \neq 1$ where $\mathsf{st}_2^{j\to i} = (\mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{a,2}^{j\to i}, \mathsf{r.ec}_{a,3}^{i\to j}, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}, \mathsf{r.ec}_{b,3}^{i\to j}, \mathsf{msg}_{2,i}, \mathsf{Trans}_1, \mathsf{c}_1^{i\to j}, \mathsf{c}_2^{j\to i}, \mathsf{c}_3^{i\to j}, \mathsf{td}_1^{j\to i}, \mathsf{nc}_i)$

- Compute $\mathsf{msg}_{3,i} \leftarrow \pi_3^{\mathsf{SM}}(\mathsf{x}_i, \mathsf{Trans}_2; \mathsf{r}_i)$.
- For each $j \in [n]$ with $j \neq i$,
  compute $\mathsf{wi}_3^{i\to j} \leftarrow \mathsf{WI}_3(\mathsf{wi}_1^{i\to j}, \mathsf{wi}_2^{j\to i}, \mathsf{st}_3^{i\to j}, \mathsf{w}_3^{i\to j})$ for the statement $\mathsf{st}_3^{i\to j} = (\mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{a,2}^{j\to i}, \mathsf{r.ec}_{a,3}^{i\to j}, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}, \mathsf{r.ec}_{b,3}^{i\to j}, \mathsf{msg}_{3,i}, \mathsf{Trans}_2, \mathsf{c}_1^{i\to j}, \mathsf{c}_2^{j\to i}, \mathsf{c}_3^{i\to j}, \mathsf{td}_1^{j\to i}) \in L_3^{i\to j}$ using witness $\mathsf{w}_3^{i\to j} = (\mathsf{x}_i, \mathsf{r}_i, \mathsf{r}_{a,\mathsf{r.ec}}^{i\to j}, \bot, \bot, \bot)$.
- Broadcast $(\mathsf{msg}_{3,i}, \mathsf{wi}_3^{i\to j})$ for all $j \in [n]$, $j \neq i$.

5. **Output Computation:**
   Let Trans_3 denote the transcript of protocol π^{SM} after round 3. P_i does the following:

   - For each $j \in [n]$ with $j \neq i$, do:
     - If $\mathsf{WI}_4(\mathsf{wi}_1^{j\to i}, \mathsf{wi}_2^{i\to j}, \mathsf{wi}_3^{j\to i}, \mathsf{st}_3^{j\to i}) \neq 1$ where $\mathsf{st}_3^{j\to i} = (\mathsf{r.ec}_{a,1}^{j\to i}, \mathsf{r.ec}_{a,2}^{i\to j}, \mathsf{r.ec}_{a,3}^{j\to i}, \mathsf{r.ec}_{a,1}^{j\to i}, \mathsf{r.ec}_{a,2}^{i\to j}, \mathsf{r.ec}_{a,3}^{j\to i}, \mathsf{msg}_{3,j}, \mathsf{Trans}_2, \mathsf{c}_1^{j\to i}, \mathsf{c}_2^{i\to j}, \mathsf{c}_3^{j\to i}, \mathsf{td}_1^{i\to j})$, broadcast an abort signal to all parties.
   - Compute output $\mathsf{y}_i \leftarrow \mathsf{OUT}(\mathsf{x}_i, \mathsf{Trans}_3; \mathsf{r}_i)$.

The correctness of the protocol follows from the correctness of all the underlying protocols.

6.2 Security Proof

In this section, we formally prove Theorem 7.
Consider an adversary $\mathcal{A}$ who corrupts t parties where $t < n$. For each party P_i, let's say that the size of input and randomness used in the protocol π^{SM} is $p(\lambda)$ for some polynomial p. That is, $|(\mathsf{x}_i, \mathsf{r}_i)| = p(\lambda)$. The strategy of the simulator Sim against a malicious adversary $\mathcal{A}$ is described below.

6.2.1 Description of Simulator

Before we describe the details, lets recall the basic strategy followed by a rewinding simulator. Sim creates a "main thread" of execution that will be actually output at the end of the simulation and a set of "look-ahead" threads that will facilitate the extraction of the adversary's input as well as some trapdoor information that is used to simulate the proofs sent by honest parties.

We construct a rewinding simulator Sim for protocol π. Simulator Sim rewinds an adversary in the second and third round of π to create a look-ahead thread. In other words, each look-ahead thread created by Sim shares the first round with the main thread, but contains different messages in the second and third rounds. Further, each look-ahead thread is terminated at the end of the third round. An important property of Sim is that it follows the honest party's strategy in all of the look ahead threads using input 0. Sim will use the adversary's messages in the third round of these look-ahead threads to extract the adversary's inputs as well as trapdoor information and then use them appropriately in the main thread to simulate the adversary's final view.

Note that as discussed in the introduction, the simulator's extraction strategy fails if the adversary aborts in the third round. In this case, the simulator just follows the honest party's strategy using input 0.

To prevent a cluttered description, we overload notation when referring to the same object in the main thread and the look-ahead threads. However, it will be clear from context which thread's object is being referred to.

The simulator's description now follows:

Step 1 - Check Abort:

1. **Round 1:**
   For each honest party P_i, Sim does the following:
   - Compute $\mathsf{msg}_{1,i} \leftarrow \mathcal{S}_1(\mathsf{r}_i)$.
   - For each $j \in [n]$ with $j \neq i$, Sim does the following:
     - Compute $\mathsf{c}_1^{i\to j} \leftarrow \mathsf{NMCom}_1(\mathsf{r}_{\mathsf{c}}^{i\to j}), \mathsf{r.ec}_{a,1}^{i\to j} \leftarrow \mathsf{R.ECom}_1(\mathsf{r}_{a,\mathsf{r.ec}}^{i\to j}), \mathsf{r.ec}_{b,1}^{i\to j} \leftarrow \mathsf{R.ECom}_1(\mathsf{r}_{b,\mathsf{r.ec}}^{i\to j})$ using random strings $\mathsf{r}_{\mathsf{c}}^{i\to j}, \mathsf{r}_{a,\mathsf{r.ec}}^{i\to j}$ and $\mathsf{r}_{b,\mathsf{r.ec}}^{i\to j}$ respectively.
     - Compute $\mathsf{td}_1^{i\to j} \leftarrow \mathsf{TDGen}_1(\mathsf{r}_{\mathsf{td}}^{i\to j})$ using a random string $\mathsf{r}_{\mathsf{td}}^{i\to j}$.
     - Generate $(\mathsf{wi}_1^{i\to j}) \leftarrow \mathsf{WI}_1(1^\lambda)$, $(\mathsf{rwi}_1^{i\to j}) \leftarrow \mathsf{RWI}_1(1^\lambda)$ and $(\mathsf{wzk}_1^{i\to j}) \leftarrow \mathsf{WZK}_1(1^\lambda)$.
   - Broadcast $(\mathsf{msg}_{1,i}, \mathsf{c}_1^{i\to j}, \mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{td}_1^{i\to j}, \mathsf{wi}_1^{i\to j}, \mathsf{rwi}_1^{i\to j}, \mathsf{wzk}_1^{i\to j})$.

2. **Round 2:**
   For each honest party P_i and for each $j \in [n]$ with $j \neq i$, Sim does the following:
   - Compute $\mathsf{c}_2^{i\to j} \leftarrow \mathsf{NMCom}_2(\mathsf{c}_1^{j\to i}), \mathsf{r.ec}_{a,2}^{i\to j} \leftarrow \mathsf{R.ECom}_2(\mathsf{r.ec}_{a,1}^{j\to i}), \mathsf{r.ec}_{b,2}^{i\to j} \leftarrow \mathsf{R.ECom}_2(\mathsf{r.ec}_{b,1}^{j\to i})$ and $\mathsf{td}_2^{i\to j} \leftarrow \mathsf{TDGen}_2(\mathsf{td}_1^{j\to i})$.
   - Compute $\mathsf{wi}_2^{i\to j} \leftarrow \mathsf{WI}_2(\mathsf{wi}_1^{j\to i}), \mathsf{rwi}_2^{i\to j} \leftarrow \mathsf{RWI}_2(\mathsf{rwi}_1^{j\to i})$ and $\mathsf{wzk}_2^{i\to j} \leftarrow \mathsf{WZK}_2(\mathsf{wzk}_1^{j\to i})$.
   - Broadcast $(\mathsf{c}_2^{i\to j}, \mathsf{r.ec}_{a,2}^{i\to j}, \mathsf{r.ec}_{b,2}^{i\to j}, \mathsf{td}_2^{i\to j}, \mathsf{wi}_2^{i\to j}, \mathsf{rwi}_2^{i\to j}, \mathsf{wzk}_2^{i\to j})$ for all j.

3. **Round 3:**
   Let Trans_1 denote the transcript after round 1 of protocol π^{SM}. For each honest P_i, Sim does the following:
   - Compute $\mathsf{msg}_{2,i} \leftarrow \mathcal{S}_2(\mathsf{Trans}_2; \mathsf{r}_i)$.
   - Compute $\mathsf{nc}_i \leftarrow \mathsf{Com}(1; \mathsf{r}_{\mathsf{nc},i})$.
   - For each $j \in [n]$ with $j \neq i$, compute:
     - $\mathsf{c}_3^{i\to j} \leftarrow \mathsf{NMCom}_3(\perp, \mathsf{c}_1^{i\to j}, \mathsf{c}_2^{j\to i}; \mathsf{r}_{\mathsf{c}}^{i\to j})$, $\mathsf{r.ec}_{a,3}^{i\to j} \leftarrow \mathsf{R.ECom}_3(0, \mathsf{r}_i, \mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{a,2}^{j\to i}; \mathsf{r}_{a,\mathsf{r.ec}}^{i\to j})$, $\mathsf{r.ec}_{b,3}^{i\to j} \leftarrow \mathsf{R.ECom}_3(\perp, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}; \mathsf{r}_{b,\mathsf{r.ec}}^{i\to j})$ using random strings $\mathsf{r}_{\mathsf{c}}^{i\to j}$, $\mathsf{r}_{a,\mathsf{r.ec}}^{i\to j}$ and $\mathsf{r}_{b,\mathsf{r.ec}}^{i\to j}$.
     - $\mathsf{td}_3^{i\to j} \leftarrow \mathsf{TDGen}_3(\mathsf{td}_1^{i\to j}, \mathsf{td}_2^{j\to i}; \mathsf{r}_{\mathsf{td}}^{i\to j})$ using randomness $\mathsf{r}_{\mathsf{td}}^{i\to j}$.
     - $\mathsf{wzk}_3^{i\to j} \leftarrow \mathsf{WZK}_3(\mathsf{wzk}_1^{i\to j}, \mathsf{wzk}_2^{j\to i}, \mathsf{st}_1^{i\to j}, \mathsf{w}_1^{i\to j})$ for the statement $\mathsf{st}_1^{i\to j} = (\mathsf{nc}_1^{i\to j}) \in L_1^{i\to j}$ using witness $\mathsf{w}_1^{i\to j} = (\mathsf{r}_{\mathsf{nc},i})$.
     - $\mathsf{rwi}_3^{i\to j} \leftarrow \mathsf{RWI}_3(\mathsf{rwi}_1^{i\to j}, \mathsf{rwi}_2^{j\to i}, \mathsf{st}_2^{i\to j}, \mathsf{w}_2^{i\to j})$ for the statement $\mathsf{st}_2^{i\to j} = (\mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{a,2}^{j\to i}, \mathsf{r.ec}_{a,3}^{i\to j}, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}, \mathsf{r.ec}_{b,3}^{i\to j}, \mathsf{msg}_{2,i}, \mathsf{Trans}_1, \mathsf{c}_1^{i\to j}, \mathsf{c}_2^{j\to i}, \mathsf{c}_3^{i\to j}, \mathsf{td}_1^{j\to i}, \mathsf{nc}_i) \in L_2^{i\to j}$ using witness $\mathsf{w}_2^{i\to j} = (0, \mathsf{r}_i, \mathsf{r}_{a,\mathsf{r.ec}}^{i\to j}, \perp, \perp, \perp, \perp)$.
   - Broadcast $(\mathsf{msg}_{2,i}, \mathsf{nc}_i, \mathsf{c}_3^{i\to j}, \mathsf{r.ec}_{a,3}^{i\to j}, \mathsf{r.ec}_{b,3}^{i\to j}, \mathsf{td}_3^{i\to j}, \mathsf{rwi}_3^{i\to j})$ for all $j \in [n]$, $j \neq i$.

4. **Abort Condition:**
   For each honest party P_i, Sim does the following:
   Let Trans_2 denote the transcript after round 2 of protocol π^{SM}. For each $j \in [n]$ with $j \neq i$, Abort if

   - $\mathsf{TDOut}(\mathsf{td}_1^{j\to i}, \mathsf{td}_2^{i\to j}, \mathsf{td}_3^{j\to i}) \neq 1$ (OR)
   - $\mathsf{WZK}_4(\mathsf{wzk}_1^{j\to i}, \mathsf{wzk}_2^{i\to j}, \mathsf{wzk}_3^{j\to i}, \mathsf{st}_1^{j\to i}) \neq 1$ where $\mathsf{st}_1^{j\to i} = (\mathsf{nc}_1^{j\to i})$. (OR)
   - $\mathsf{RWI}_4(\mathsf{rwi}_1^{i\to j}, \mathsf{rwi}_2^{i\to j}, \mathsf{rwi}_3^{j\to i}, \mathsf{st}_2^{j\to i}) \neq 1$ where $\mathsf{st}_2^{j\to i} = (\mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{a,2}^{j\to i}, \mathsf{r.ec}_{a,3}^{i\to j}, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}, \mathsf{r.ec}_{b,3}^{i\to j}, \mathsf{msg}_{2,i}, \mathsf{Trans}_1, \mathsf{c}_1^{i\to j}, \mathsf{c}_2^{j\to i}, \mathsf{c}_3^{i\to j}, \mathsf{td}_1^{j\to i}, \mathsf{nc}_i)$

If Sim doesn't abort at this point, we will say that the "Check Abort" step succeeded.

Step 2 - Rewinding:

- Sim now rewinds back to the end of round 1 of the protocol. Then, Sim creates a set of T (defined later) look-ahead threads that run only rounds 2 and 3 of the protocol in the following manner:

  5. **Round 2:**
     In every look-ahead thread, for each honest party P_i and for each $j \in [n]$ with $j \neq i$, Sim does exactly as done in round 2 of step 1.
  6. **Round 3:**
     In every look-ahead thread, for each honest party P_i and for each $j \in [n]$ with $j \neq i$, Sim does exactly as done in round 3 of step 1.

- For each thread above, define it to be "Bad" if for any honest party P_i and any $j \in [n]$ with $j \neq i$:
  - $\mathsf{TDOut}(\mathsf{td}_1^{j\to i}, \mathsf{td}_2^{i\to j}, \mathsf{td}_3^{j\to i}) \neq 1$ (OR)
  - $\mathsf{WZK}_4(\mathsf{wzk}_1^{j\to i}, \mathsf{wzk}_2^{i\to j}, \mathsf{wzk}_3^{j\to i}, \mathsf{st}_1^{j\to i}) \neq 1$ where $\mathsf{st}_1^{j\to i} = (\mathsf{nc}_1^{j\to i})$. (OR)
  - $\mathsf{RWI}_4(\mathsf{rwi}_1^{i\to j}, \mathsf{rwi}_2^{i\to j}, \mathsf{rwi}_3^{j\to i}, \mathsf{st}_2^{j\to i}) \neq 1$ where $\mathsf{st}_2^{j\to i} = (\mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{a,2}^{j\to i}, \mathsf{r.ec}_{a,3}^{i\to j}, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}, \mathsf{r.ec}_{b,3}^{i\to j}, \mathsf{msg}_{2,i}, \mathsf{Trans}_1, \mathsf{c}_1^{i\to j}, \mathsf{c}_2^{j\to i}, \mathsf{c}_3^{i\to j}, \mathsf{td}_1^{j\to i}, \mathsf{nc}_i)$
- The number of threads T created is such that at least $(12 \cdot \lambda)$ "Good" threads exist. That is, Sim keeps running till it receives $(12 \cdot \lambda)$ "Good" threads.

Step 3 - Input Extraction:
Sim does the following:

- Run the trapdoor extractor using the trapdoor generation messages of 3 "Good" threads to extract a valid trapdoor. That is, compute $\mathsf{t}_j \leftarrow \mathsf{TDExt}(\mathsf{td}_1^{j\to i}, \{\mathsf{td}_2^{i\to j}, \mathsf{td}_3^{j\to i}\})$ where the set denotes the pair of values from all the threads.

- Run the extractor $\mathsf{Ext}_{\mathsf{R.ECom}}$ on the first extractable commitment messages using 5 "Good" threads. That is, compute $(\mathsf{x}_{a,j}, \mathsf{r}_{a,j}) \leftarrow \mathsf{Ext}_{\mathsf{R.ECom}}(\mathsf{r.ec}_{a,1}^{j\to i}, \{\mathsf{r.ec}_{a,2}^{i\to j}, \mathsf{r.ec}_{a,3}^{j\to i}\})$ where the set denotes the pair of values from all the threads. Similarly, using the second extractable commitment, compute $(\mathsf{x}_{b,j}, \mathsf{r}_{b,j})$. Check which of them are consistent with the messages $\mathsf{msg}_{1,j}, \mathsf{msg}_{2,j}$ in the underlying semi-malicious protocol and set that pair to be $(\mathsf{x}_j, \mathsf{r}_j)$.

- Output "Special Abort" if any of the above two steps fail.

Step 4 - Query to Ideal Functionality:

- Sim queries the ideal functionality with the set of values $\{\mathsf{x}_j\}$ where x_j is the input of the each adversarial party P_j extracted in the previous step.
- Sim receives output y from the ideal functionality.
- Let R denote the set of all $\{\mathsf{x}_j, \mathsf{r}_j\}$ extracted in the previous step.

Step 5 - Abort Probability Estimation:
Set $\epsilon' = \frac{12\lambda}{T}$ as the probability with which the adversary doesn't abort.

Step 6 - Continuing Main Thread:
First, Sim sets a counter value to 0. Sim now rewinds back to the end of round 1 and continues execution on the main thread. Note that this step also involves rewinding as elaborated below.

7. **Round 2:**
   Run exactly as before - i.e as done in step 1 and in each of the look-ahead threads.

8. **Round 3:**
   Now, the only difference from the look-ahead threads is that in the non-malleable commitment execution between honest party i and adversarial party j, Sim commits to the trapdoor t_j (extracted above). Let Trans_1 denote the transcript after round 1 of protocol π^{SM}. For each honest P_i, Sim does the following:

   - Compute $\mathsf{msg}_{2,i} \leftarrow \mathcal{S}_2(\mathsf{Trans}_1; \mathsf{r}_i)$.
   - Compute $\mathsf{nc}_i \leftarrow \mathsf{Com}(1; \mathsf{r}_{\mathsf{nc},i})$.
   - For each $j \in [n]$ with $j \neq i$, compute:
     - Compute $\mathsf{c}_3^{i\to j} \leftarrow \mathsf{NMCom}_3(\mathsf{t}_j, \mathsf{c}_1^{i\to j}, \mathsf{c}_2^{j\to i}; \mathsf{r}_{\mathsf{c}}^{i\to j})$, $\mathsf{r.ec}_{a,3}^{i\to j} \leftarrow \mathsf{R.ECom}_3(0, \mathsf{r}_i, \mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{a,2}^{j\to i}; \mathsf{r}_{a,\mathsf{r.ec}}^{i\to j})$, $\mathsf{r.ec}_{b,3}^{i\to j} \leftarrow \mathsf{R.ECom}_3(\perp, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}; \mathsf{r}_{b,\mathsf{r.ec}}^{i\to j})$ using random strings $\mathsf{r}_{\mathsf{c}}^{i\to j}$, $\mathsf{r}_{a,\mathsf{r.ec}}^{i\to j}$ and $\mathsf{r}_{b,\mathsf{r.ec}}^{i\to j}$.
     - Compute $\mathsf{td}_3^{i\to j} \leftarrow \mathsf{TDGen}_3(\mathsf{td}_1^{i\to j}, \mathsf{td}_2^{j\to i}; \mathsf{r}_{\mathsf{td}}^{i\to j})$ using randomness $\mathsf{r}_{\mathsf{td}}^{i\to j}$.
     - $\mathsf{wzk}_3^{i\to j} \leftarrow \mathsf{WZK}_3(\mathsf{wzk}_1^{i\to j}, \mathsf{wzk}_2^{j\to i}, \mathsf{st}_1^{i\to j}, \mathsf{w}_1^{i\to j})$ for the statement $\mathsf{st}_1^{i\to j} = (\mathsf{nc}_1^{i\to j}) \in L_1^{i\to j}$ using witness $\mathsf{w}_1^{i\to j} = (\mathsf{r}_{\mathsf{nc},i})$.
     - $\mathsf{rwi}_3^{i\to j} \leftarrow \mathsf{RWI}_3(\mathsf{rwi}_1^{i\to j}, \mathsf{rwi}_2^{j\to i}, \mathsf{st}_2^{i\to j}, \mathsf{w}_2^{i\to j})$ for the statement $\mathsf{st}_2^{i\to j} = (\mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{a,2}^{j\to i}, \mathsf{r.ec}_{a,3}^{i\to j}, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}, \mathsf{r.ec}_{b,3}^{i\to j}, \mathsf{msg}_{2,i}, \mathsf{Trans}_1, \mathsf{c}_1^{i\to j}, \mathsf{c}_2^{j\to i}, \mathsf{c}_3^{i\to j}, \mathsf{td}_1^{j\to i}, \mathsf{nc}_i) \in L_2^{i\to j}$ using witness $\mathsf{w}_2^{i\to j} = (0, \mathsf{r}_i, \mathsf{r}_{a,\mathsf{r.ec}}^{i\to j}, \perp, \perp, \perp, \perp)$.
   - Broadcast $(\mathsf{msg}_{2,i}, \mathsf{nc}_i, \mathsf{c}_3^{i\to j}, \mathsf{r.ec}_{a,3}^{i\to j}, \mathsf{r.ec}_{b,3}^{i\to j}, \mathsf{td}_3^{i\to j}, \mathsf{rwi}_3^{i\to j})$ for all $j \in [n]$, $j \neq i$.

9. **Abort Condition:**
   Let Trans_2 denote the transcript after round 2 of protocol π^{SM}. For each honest party P_i, Sim does the following:

   - For each $j \in [n]$ with $j \neq i$, increase the counter value by 1 if
     - $\mathsf{TDOut}(\mathsf{td}_1^{j\to i}, \mathsf{td}_2^{i\to j}, \mathsf{td}_3^{j\to i}) \neq 1$ (OR)
     - $\mathsf{WZK}_4(\mathsf{wzk}_1^{j\to i}, \mathsf{wzk}_2^{i\to j}, \mathsf{wzk}_3^{j\to i}, \mathsf{st}_1^{j\to i}) \neq 1$ where $\mathsf{st}_1^{j\to i} = (\mathsf{nc}_1^{j\to i})$. (OR)
     - $\mathsf{RWI}_4(\mathsf{rwi}_1^{i\to j}, \mathsf{rwi}_2^{j\to i}, \mathsf{rwi}_3^{j\to i}, \mathsf{st}_2^{j\to i}) \neq 1$ where $\mathsf{st}_2^{j\to i} = (\mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{a,2}^{j\to i}, \mathsf{r.ec}_{a,3}^{i\to j}, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}, \mathsf{r.ec}_{b,3}^{i\to j}, \mathsf{msg}_{2,i}, \mathsf{Trans}_1, \mathsf{c}_1^{i\to j}, \mathsf{c}_2^{j\to i}, \mathsf{c}_3^{i\to j}, \mathsf{td}_1^{j\to i}, \mathsf{nc}_i)$

- If Sim's running time equals 2^λ, Abort.
- If the counter value was not increased in this round, (i.e, none of the abort conditions were true), continue to Step 7.
- Else, if counter value less than $\frac{\lambda^2}{\epsilon'}$, rewind back to the beginning of round 2 in this step (step 6).

Step 7 - Continuing Main Thread:

9. **Round 4:**
   Recall that Sim doesn't continue the look-ahead threads (used in extraction) from this round. Let Trans_2 denote the transcript after round 2 of protocol π^{SM}. In the main thread, for each honest party P_i, Sim does the following:
   - Compute $\mathsf{msg}_{3,i} \leftarrow \mathcal{S}_3(\mathsf{y}, \mathsf{R}, \mathsf{Trans}_2, i)$.
   - For each $j \in [n]$ with $j \neq i$,
     Compute $\mathsf{wi}_3^{i\to j} \leftarrow \mathsf{WI}_3(\mathsf{wi}_1^{i\to j}, \mathsf{wi}_2^{j\to i}, \mathsf{st}_3^{i\to j}, \mathsf{w}_3^{i\to j})$ for the statement $\mathsf{st}_3^{i\to j} = (\mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{a,2}^{j\to i}, \mathsf{r.ec}_{a,3}^{i\to j}, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}, \mathsf{r.ec}_{b,3}^{i\to j}, \mathsf{msg}_{3,i}, \mathsf{Trans}_2, \mathsf{c}_1^{i\to j}, \mathsf{c}_2^{j\to i}, \mathsf{c}_3^{i\to j}, \mathsf{td}_1^{j\to i}) \in L_3^{i\to j}$ using witness $\mathsf{w}_3^{i\to j} = (\bot, \bot, \bot, \bot, \mathsf{t}_j, \mathsf{r}_{\mathsf{c}}^{i\to j})$.
   - Broadcast $(\mathsf{msg}_{3,i}, \mathsf{wi}_3^{i\to j})$ for all j.

10. **Output Computation:**
    Let Trans_3 denote the transcript of protocol π^{SM} after round 3. In the main thread, for each honest party P_i and every $j \in [n]$ with $j \neq i$, Sim does the following:
    - Abort if $\mathsf{WI}_4(\mathsf{wi}_1^{j\to i}, \mathsf{wi}_2^{i\to j}, \mathsf{wi}_3^{j\to i}, \mathsf{st}_3^{j\to i}) \neq 1$ where $\mathsf{st}_3^{j\to i} = (\mathsf{r.ec}_{a,1}^{j\to i}, \mathsf{r.ec}_{a,2}^{i\to j}, \mathsf{r.ec}_{a,3}^{j\to i}, \mathsf{r.ec}_{a,1}^{j\to i}, \mathsf{r.ec}_{a,2}^{i\to j}, \mathsf{r.ec}_{a,3}^{j\to i}, \mathsf{msg}_{3,j}, \mathsf{Trans}_2, \mathsf{c}_1^{j\to i}, \mathsf{c}_2^{i\to j}, \mathsf{c}_3^{j\to i}, \mathsf{td}_1^{i\to j})$. In particular, send a global abort signal to all parties so that everyone aborts.

    Finally, instruct the ideal functionality to deliver output to the honest parties.

We now prove that the simulator is an expected PPT machine.

Claim 7. *Simulator* Sim *runs in expected time that is polynomial in* λ.

Proof. Let's analyze the running time of each step of the simulation strategy. Clearly, step 1 takes only $\mathsf{poly}(\lambda)$ time for some polynomial.

Let ϵ be the probability with which the adversary doesn't abort. That is, Sim proceeds to step 2 only with probability ϵ. Now, since the probability of the adversary not aborting is ϵ, the expected number of threads to be run by the simulator to get one non-aborting transcript is $\frac{1}{\epsilon}$. Therefore, the expected total number of threads created in step 2 is $\frac{12\cdot\lambda}{\epsilon}$ and each thread takes only $\mathsf{poly}(\lambda)$ time.

In step 3, the extractors TDExt and $\mathsf{Ext}_{\mathsf{R.ECom}}$ are PPT machines. Steps 4 and 5 are trivially polynomial time.

As shown in [GK96, Lin17], the probability that the estimate ϵ' computed in step 5 is not within a factor of 2 of ϵ is at most 2^λ. An exact computation of how to achieve this exact bound using Chernoff bounds can be found in [HL10], Section 6.5.3. (which also explains why we chose to run step 2 till we get $12 \cdot \lambda$ non-aborting transcripts). Therefore, the number of threads created in step 6 is at most $\frac{\lambda^2}{\epsilon}$ (ignoring the constant factor). Note that step 6 might still take time 2^λ but this happens only when the estimate of ϵ' is incorrect: that is, when ϵ' is not within a constant factor of ϵ and this happens only with probability $2^{-\lambda}$.

Finally, it is easy to see that step 7 runs in $\mathsf{poly}(\lambda)$.
Therefore, we can bound the overall running time by :

$$\mathsf{T}_{\mathsf{Sim}} = \mathsf{poly}(\lambda) + \mathsf{poly}(\lambda) \cdot \epsilon \left(\frac{12 \cdot \lambda}{\epsilon} + \left(1 - \frac{1}{2^\lambda}\right) \cdot \frac{\lambda^2}{\epsilon} + \left(\frac{1}{2^\lambda}\right) \cdot 2^\lambda \right)$$

$$\leq \mathsf{poly}(\lambda)$$

for some polynomial and this concludes the analysis. □

6.2.2 Hybrids

We now show that the above simulation strategy is successful against all malicious PPT adversaries. That is, the view of the adversary along with the output of the honest parties is computationally indistinguishable in the real and ideal worlds. We will show this via a series of computationally indistinguishable hybrids where the first hybrid Hyb_0 corresponds to the real world and the last hybrid Hyb_{11} corresponds to the ideal world.

First, assume by contradiction that there exists an adversary $\mathcal{A}$ that can distinguish the real and ideal worlds with some non-negligible probability μ. We will use this value μ in the hybrids.

- Hyb_0 **- Real World:** In this hybrid, consider a simulator $\mathsf{Sim}_{\mathsf{Hyb}}$ that plays the role of the honest parties. This corresponds to the real world experiment.

- Hyb_1 **- Aborting Scenario:** In this hybrid, $\mathsf{Sim}_{\mathsf{Hyb}}$ first runs the "Check Abort" step - that is, step 1 in the description of Sim to check if the adversary aborts. That is, $\mathsf{Sim}_{\mathsf{Hyb}}$ performs the first 3 rounds of the protocol using the honest parties' strategy but using input 0.

  Suppose the adversary doesn't cause an abort - that is, the "Check Abort" step succeeded. Then, $\mathsf{Sim}_{\mathsf{Hyb}}$ rewinds back to the end of round 1 of the protocol and performs exactly as in Hyb_0. In particular, $\mathsf{Sim}_{\mathsf{Hyb}}$ no longer uses input 0 in the main thread that was done in the "Check Abort" step and instead continues using the honest parties' actual inputs. Additionally, if in the main thread, $\mathsf{Sim}_{\mathsf{Hyb}}$ receives an Abort at the end of round 3, it rewinds back to the end of round 1 and runs the main thread again. This process happens $\frac{1}{\mu}$ times. Observe that $\mathsf{Sim}_{\mathsf{Hyb}}$ runs in polynomial time because, by assumption, μ was non-negligible. The same argument holds for the subsequent hybrids as well.

- Hyb_2 **- Input Extraction:** In this hybrid, suppose the "Check Abort" step succeeds, then, $\mathsf{Sim}_{\mathsf{Hyb}}$ creates a set of look-ahead threads (or rewound threads) that run only rounds 2 and 3 of the protocol till it extracts 5 "Good" threads. In all the threads (main and look-aheads), $\mathsf{Sim}_{\mathsf{Hyb}}$ plays the role of the honest parties exactly as in Hyb_0. Additionally, $\mathsf{Sim}_{\mathsf{Hyb}}$ also runs the "Input Extraction" phase and "Query to Ideal Functionality" phase described in steps 3 and 4 of the description of Sim. That is, it extracts the adversary's input, randomness and a set of valid trapdoors and also, queries the ideal functionality using the adversary's inputs to receive the protocol output. Note that in the main thread, $\mathsf{Sim}_{\mathsf{Hyb}}$ continues to perform exactly as in Hyb_1 by rewinding $\frac{1}{\mu}$ times.

- Hyb_3 **- Changing NMCom:** In the main thread, for each honest party P_i and every malicious party P_j, $\mathsf{Sim}_{\mathsf{Hyb}}$ does the following: in round 3, compute $\mathsf{c}_3^{i \to j} = \mathsf{NMCom}_3(\mathsf{t}_j, \mathsf{c}_1^{i \to j}, \mathsf{c}_2^{j \to i}; \mathsf{r}_{\mathsf{c}}^{i \to j})$ where t_j is a valid trapdoor extracted as in the previous hybrid.
  That is, in the main thread, the simulator now commits to a valid trapdoor of the adversary's trapdoor generation messages, using the non-malleable commitment scheme.

Note that this happens in each of the potentially rewound main threads till $\mathsf{Sim}_{\mathsf{Hyb}}$ receives a non-aborting transcript after round 3 or the number of rewinds reaches $\frac{1}{\mu}$. Once again, recall that these rewinds are different from the look-ahead threads that were created for extraction. The same argument applies to each subsequent hybrid as well when we make changes on the main thread.

- Hyb_4 **- Switching Rewinding WI proofs:** In the main thread, for each honest party P_i and every malicious party P_j, $\mathsf{Sim}_{\mathsf{Hyb}}$ does the following: in round 3, compute $\mathsf{rwi}_2^{i\to j} \leftarrow \mathsf{RWI}_2(\mathsf{rwi}_1^{j\to i}, \mathsf{st}_2^{i\to j}, \mathsf{w}_2^{i\to j})$ for the statement $\mathsf{st}_2^{i\to j} = (\mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{a,2}^{j\to i}, \mathsf{r.ec}_{a,3}^{i\to j}, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}, \mathsf{r.ec}_{b,3}^{i\to j}, \mathsf{msg}_{2,i}, \mathsf{Trans}_1, \mathsf{c}_1^{i\to j}, \mathsf{c}_2^{j\to i}, \mathsf{c}_3^{i\to j}, \mathsf{td}_1^{j\to i}, \mathsf{nc}_i) \in L_2^{i\to j}$ using witness $\mathsf{w}_2^{i\to j} = (\bot, \bot, \bot, \bot, \mathsf{t}_j, \mathsf{r}_{\mathsf{c}}^{i\to j}, \bot)$.
  That is, in the main thread, the simulator now uses the witness for the non-malleable commitment to generate proofs in round 3.

- Hyb_5 **- Switching WI proofs:** In the main thread, for each honest party P_i and every malicious party P_j, $\mathsf{Sim}_{\mathsf{Hyb}}$ does the following: in round 4, compute $\mathsf{wi}_3^{i\to j} \leftarrow \mathsf{WI}_3(\mathsf{wi}_1^{i\to j}, \mathsf{wi}_2^{j\to i}, \mathsf{st}_3^{i\to j}, \mathsf{w}_3^{i\to j})$ for the statement $\mathsf{st}_3^{i\to j} = (\mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{a,2}^{j\to i}, \mathsf{r.ec}_{a,3}^{i\to j}, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}, \mathsf{r.ec}_{b,3}^{i\to j}, \mathsf{msg}_{3,i}, \mathsf{Trans}_2, \mathsf{c}_1^{i\to j}, \mathsf{c}_2^{j\to i}, \mathsf{c}_3^{i\to j}, \mathsf{td}_1^{j\to i}) \in L_3^{i\to j}$ using witness $\mathsf{w}_3^{i\to j} = (\bot, \bot, \bot, \bot, \mathsf{t}_j, \mathsf{r}_{\mathsf{c}}^{i\to j})$.
  That is, in the main thread, the simulator now uses the witness for the non-malleable commitment to generate proofs in rounds 4.

- Hyb_6 **- Changing ExtCom:** In the main thread, for each honest party P_i, $\mathsf{Sim}_{\mathsf{Hyb}}$ does the following: in round 3, compute $\mathsf{r.ec}_{a,3}^{i\to j} = \mathsf{R.ECom}_3(0, \mathsf{r}_i, \mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{a,2}^{j\to i}; \mathsf{r}_{a,\mathsf{r.ec}}^{i\to j})$.
  That is, in the main thread, the simulator now commits to input 0 using the extractable commitment scheme.

- Hyb_7 **- Simulate π^{SM}:** In the main thread, for each honest party P_i, $\mathsf{Sim}_{\mathsf{Hyb}}$ does the following:
  - in round 1, compute $\mathsf{msg}_{1,i} \leftarrow \mathcal{S}_1(\mathsf{r}_i)$. The description of $\mathsf{Sim}_{\mathsf{Hyb}}$ now corresponds to the ideal world simulator Sim.
  - In round 3, compute $\mathsf{msg}_{2,i} \leftarrow \mathcal{S}_2(\mathsf{Trans}_1; \mathsf{r}_i)$ where y is the output from the ideal functionality, R denotes the set of all $\{\mathsf{x}_j, \mathsf{r}_j\}$ extracted and Trans_2 denotes the transcript of protocol π^{SM} after round 2.
  - in round 4, compute $\mathsf{msg}_{3,i} \leftarrow \mathcal{S}_3(\mathsf{y}, \mathsf{R}, \mathsf{Trans}_2, i)$ where y is the output from the ideal functionality, R denotes the set of all $\{\mathsf{x}_j, \mathsf{r}_j\}$ extracted and Trans_2 denotes the transcript of protocol π^{SM} after round 2.

- Hyb_8 **- Switching back Rewinding WI proofs:** In the main thread, for each honest party P_i and every malicious party P_j, $\mathsf{Sim}_{\mathsf{Hyb}}$ does the following: in round 3, compute $\mathsf{rwi}_2^{i\to j} \leftarrow \mathsf{RWI}_2(\mathsf{rwi}_1^{j\to i}, \mathsf{st}_2^{i\to j}, \mathsf{w}_2^{i\to j})$ for the statement $\mathsf{st}_2^{i\to j} = (\mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{a,2}^{j\to i}, \mathsf{r.ec}_{a,3}^{i\to j}, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}, \mathsf{r.ec}_{b,3}^{i\to j}, \mathsf{msg}_{2,i}, \mathsf{Trans}_1, \mathsf{c}_1^{i\to j}, \mathsf{c}_2^{j\to i}, \mathsf{c}_3^{i\to j}, \mathsf{td}_1^{j\to i}, \mathsf{nc}_i) \in L_2^{i\to j}$ using witness $\mathsf{w}_2^{i\to j} = (0, \mathsf{r}_i, \mathsf{r}_{a,\mathsf{r.ec}}^{i\to j}, \bot, \bot, \bot, \bot)$.
  That is, in the main thread, the simulator now uses the original witness to generate proofs in rounds 3.

- Hyb_9 **- Extra look-ahead threads:** In this hybrid, $\mathsf{Sim}_{\mathsf{Hyb}}$ creates another set of look-ahead threads that once again, only run rounds 2 and 3 exactly as in the previous hybrid. For clarity of exposition, let's call the first set of look-ahead threads as T_1 and the second set as T_2. As before, $\mathsf{Sim}_{\mathsf{Hyb}}$ extracts values $\{\mathsf{t}_j, \mathsf{x}_j, \mathsf{r}_j\}$ using the first set of threads T_1.

$\mathsf{Sim}_{\mathsf{Hyb}}$ also runs the "input extraction phase" using this second set of look-ahead threads. That is, $\mathsf{Sim}_{\mathsf{Hyb}}$ also extracts values $\{\mathsf{t}_j^*, \mathsf{x}_j^*, \mathsf{r}_j^*\}$ using the second set of threads T_2 and outputs "Special Abort" if this extraction is not successful. Further, $\mathsf{Sim}_{\mathsf{Hyb}}$ outputs "Special Abort 2" if there $\exists\ j$ such that $(\mathsf{x}_j, \mathsf{r}_j) \neq (\mathsf{x}_j^*, \mathsf{r}_j^*)$ in any of the second set of look-ahead threads. $\mathsf{Sim}_{\mathsf{Hyb}}$ continues to use the first set of extracted values in the rest of the protocol.

- Hyb_{10} **- Changes to extra set:** On each thread in T_2, $\mathsf{Sim}_{\mathsf{Hyb}}$ performs the changes outlined in hybrids 3 to 8. That is, the changes that were made to the main thread in hybrids 3 through 8 are now performed in each of these look-ahead threads in T_2 one at a time. This hybrid can be expanded into 6 hybrids for each thread in T_2.

- Hyb_{11} **- Stop set 1 of look-aheads:** $\mathsf{Sim}_{\mathsf{Hyb}}$ now stops executing the first set of look-ahead threads T_1 and uses the extracted values $\{\mathsf{t}_j^*, \mathsf{x}_j^*, \mathsf{r}_j^*\}$ from the second set. Therefore, at this point, $\mathsf{Sim}_{\mathsf{Hyb}}$ only runs the main thread and the set T_2 consisting of the look-ahead threads.

  Additionally, at this point, $\mathsf{Sim}_{\mathsf{Hyb}}$ doesn't rewind the main thread $\frac{1}{\mu}$ times. Instead, $\mathsf{Sim}_{\mathsf{Hyb}}$ first estimates the probability of the adversary not aborting - ϵ' as done in step 5 in the description of Sim and then runs the rewound main thread for $\mathsf{max}(2^\lambda, \frac{\lambda^2}{\epsilon'})$ time as in the ideal world. This hybrid exactly corresponds to the ideal world.

6.2.3 Indistinguishability of hybrids

Throughout the sequence of hybrids, starting with Hyb_1, we will maintain the following invariant which will be useful to argue the proof of indistinguishability.

Definition 6 (Invariant)**.** *Consider any malicious party P_j and any honest party P_i. $\mathsf{td}_1^{i\to j}$ denotes the first message of the trapdoor generation protocol with P_i as the trapdoor generator. The values $(\mathsf{c}_1^{j\to i}, \mathsf{c}_2^{i\to j}, \mathsf{c}_3^{j\to i})$ denote the messages of the non-malleable commitment with P_j as the creator. Consider the following event E which occurs if $\exists(\mathsf{t}_i, \mathsf{r}_\mathsf{c}^{j\to i})$ such that:*

- $\mathsf{c}_1^{j\to i} = \mathsf{NMCom}_1(\mathsf{r}_\mathsf{c}^{j\to i})$ *(AND)*
- $\mathsf{c}_3^{j\to i} = \mathsf{NMCom}_3(\mathsf{t}_i, \mathsf{c}_1^{j\to i}, \mathsf{c}_2^{i\to j}; \mathsf{r}_\mathsf{c}^{j\to i})$. *(AND)*
- $\mathsf{TDValid}(\mathsf{td}_1^{i\to j}, \mathsf{t}_i) = 1$.

That is, the event E occurs if the adversary P_j, using the non-malleable commitment, commits to a valid trapdoor t_i for the trapdoor generation messages of the honest party P_i.

The invariant is : $\Pr[$ *Event* E *occurs in any thread* $] \leq \mathsf{negl}(\lambda)$.

Claim 8. *Assuming the "1-rewinding security" of the trapdoor generation protocol* TDGen *and the existence of an extractor* $\mathsf{Ext}_{\mathsf{NMCom}}$ *for the non-malleable commitment scheme* NMCom*, the invariant holds in* Hyb_0*.*

Proof. We will prove this by contradiction. Assume that the invariant doesn't hold in Hyb_0. That is, there exists an adversary $\mathcal{A}$ such that for some honest party P_i^* and malicious party P_j^*, $\mathcal{A}$ causes event E to occur with non-negligible probability. We will use this adversary to design an adversary $\mathcal{A}_{\mathsf{TDGen}}$ that breaks the "1-rewinding security" of the trapdoor generation protocol TDGen as defined in Section 4 with non-negligible probability.

$\mathcal{A}_{\mathsf{TDGen}}$ interacts with a challenger $\mathcal{C}_{\mathsf{TDGen}}$ and receives a first round message td_1 corresponding to the protocol TDGen. $\mathcal{A}_{\mathsf{TDGen}}$ performs the role of $\mathsf{Sim}_{\mathsf{Hyb}}$ in its interaction with $\mathcal{A}$ exactly as done in Hyb_0. $\mathsf{Sim}_{\mathsf{Hyb}}$ picks an honest party P_i uniformly at random and a malicious party P_j uniformly at random. Now, in round 1 of protocol π, $\mathcal{A}_{\mathsf{TDGen}}$ sets $\mathsf{td}_1^{i\to j}$ as td_1 received from $\mathcal{C}_{\mathsf{TDGen}}$. On receiving a value $\mathsf{td}_2^{j\to i}$ from $\mathcal{A}$ in round 2, $\mathcal{A}_{\mathsf{TDGen}}$ forwards this message to $\mathcal{C}_{\mathsf{TDGen}}$ as its second round message for the protocol TDGen. $\mathcal{A}_{\mathsf{TDGen}}$ receives td_3 from $\mathcal{C}_{\mathsf{TDGen}}$ which is set as $\mathsf{td}_3^{i\to j}$ in its interaction with $\mathcal{A}$. $\mathcal{A}_{\mathsf{TDGen}}$ continues with the rest of protocol π exactly as in Hyb_0 upto round 3. At this point, $\mathcal{A}_{\mathsf{TDGen}}$ rewinds the adversary $\mathcal{A}$ back to the beginning of round 2. To be consistent with our earlier terminology, this can be interpreted as follows: in the security proof, $\mathcal{A}_{\mathsf{TDGen}}$ creates a look-ahead thread that runs only rounds 2 and 3 of protocol π. Note that this look-ahead thread exists only in the proof of the invariant and not in the description of Hyb_0. As in the main thread, $\mathcal{A}_{\mathsf{TDGen}}$ forwards the adversary's message $\mathsf{td}_2^{j\to i}$ from $\mathcal{A}$ in round 2 to $\mathcal{C}_{\mathsf{TDGen}}$ and receives td_3 from $\mathcal{C}_{\mathsf{TDGen}}$ which is set as $\mathsf{td}_3^{i\to j}$ in its interaction with $\mathcal{A}$. $\mathcal{A}_{\mathsf{TDGen}}$ continues with the rest of protocol π exactly as in Hyb_0.

Now, $\mathcal{A}_{\mathsf{TDGen}}$ runs the extractor $\mathsf{Ext}_{\mathsf{NMCom}}$ of the non-malleable commitment scheme using the messages in both the threads that correspond to the non-malleable commitment from malicious party P_j to honest party P_i. Let the output of $\mathsf{Ext}_{\mathsf{NMCom}}$ be t^*. $\mathcal{A}_{\mathsf{TDGen}}$ outputs t^* as a valid trapdoor to $\mathcal{C}_{\mathsf{TDGen}}$.

Let's analyze why this works. We know that the invariant doesn't hold so there exists honest party P_i^* and malicious party P_j^* such that event E doesn't hold. That is, the adversary P_j^*, using the non-malleable commitment, commits to a valid trapdoor t_i^* for the trapdoor generation messages of the honest party P_i^* with non-negligible probability ϵ. With probability $\frac{1}{n^2}$ where n is the total number of parties, this corresponds to honest party P_i and malicious party P_j picked randomly by $\mathcal{A}_{\mathsf{TDGen}}$. Therefore, with non-negligible probability $\frac{\epsilon}{n^2}$, the adversary P_j, using the non-malleable commitment, commits to a valid trapdoor t_i^* for the trapdoor generation messages of the honest party P_i. Therefore, by definition, given the messages of the non-malleable commitment in 2 threads, the extractor $\mathsf{Ext}_{\mathsf{NMCom}}$ is successful with non-negligible probability ϵ'. Therefore, with non-negligible probability $\frac{\epsilon*\epsilon'}{n^2}$, $\mathcal{A}_{\mathsf{TDGen}}$ outputs t^* as a valid trapdoor to $\mathcal{C}_{\mathsf{TDGen}}$ which breaks the security of the trapdoor generation protocol TDGen. Thus, it must be the case that the invariant holds in Hyb_0. □

Claim 9. *If the "Check Abort" step succeeds, the invariant holds in* Hyb_1.

Proof. Since there is no difference in the main thread between Hyb_0 and Hyb_1 when the "Check Abort" step succeeds, the invariant continues to hold true. □

Claim 10. *Assuming the hiding property of the commitment scheme* Com*, the weak ZK property of the scheme* WZK*, the witness indistinguishability property of the scheme* WI*, the security of the protocol* π^{SM}*,* Hyb_0 *is indistinguishable from* Hyb_1.

Proof. In Hyb_0, $\mathsf{Sim}_{\mathsf{Hyb}}$ runs the protocol once using the honest strategy. In Hyb_1, if the "Check Abort" step succeeded, $\mathsf{Sim}_{\mathsf{Hyb}}$ runs the protocol using the honest strategy $\frac{1}{\mu}$ times where μ is the adversary's distinguishing advantage in the overall experiment. Suppose the adversary doesn't abort with probability greater than $\frac{1}{\mu}$, then, by Markov's inequality, except with negligible probability, the adversary's view in Hyb_1 consists of a thread of execution identically distributed to Hyb_0.

Therefore, the only difference between the two hybrids is if the adversary causes $\mathsf{Sim}_{\mathsf{Hyb}}$ to abort at the end of round 3 with probability greater than $\frac{1}{\mu}$- that is, the "Check Abort" step doesn't succeed in Hyb_1. In that case, in Hyb_0, $\mathsf{Sim}_{\mathsf{Hyb}}$ uses the honest parties' inputs to run the protocol while in Hyb_1, $\mathsf{Sim}_{\mathsf{Hyb}}$ runs the protocol using input 0 for every honest party. We show in

Appendix C via a complicated sequence of sub-hybrids that these two hybrids are indistinguishable even in this case.

□

We now get back to proving security of the main simulation.

Claim 11. *If the "Check Abort" step succeeds, the invariant holds in* Hyb_2.

Proof. There is no difference in the main thread between Hyb_1 and Hyb_2 when the "Check Abort" step succeeds. Also, each look-ahead thread is identical to the main thread and so the invariant continues to hold true. □

Claim 12. *Assuming soundness of the argument systems* $\mathsf{WI}, \mathsf{RWI}$ *and* WZK*, the existence of an extractor* Ext *for the extractable commitment scheme* $\mathsf{R.ECom}$ *and the existence of the trapdoor extractor* TDExt *for the trapdoor generation protocol* TDGen*,* Hyb_1 *is indistinguishable from* Hyb_2.

Proof. Observe that the only difference between the two hybrids is that the simulator outputs "Special Abort" in the input extraction phase in Hyb_2. To prove that the two hybrids are indistinguishable, we will now show that the $\mathsf{Pr}[\ \mathsf{Sim}_{\mathsf{Hyb}}$ outputs "Special Abort" $\ ] \leq \mathsf{negl}(\lambda)$ in Hyb_2. $\mathsf{Sim}_{\mathsf{Hyb}}$ outputs "Special Abort" only if either of the algorithms TDExt or $\mathsf{Ext}_{\mathsf{R.ECom}}$ fail.

Recall that we denote by ϵ the probability with which the adversary causes the "Abort Check" step to succeed.

Case 1: $\epsilon \leq \mathsf{negl}(\lambda)$

In this case, with probability $\geq (1 - \mathsf{negl}(\lambda))$, the adversary causes the "Abort Check" step to fail and hence $\mathsf{Sim}_{\mathsf{Hyb}}$ outputs "Special Abort" only with probability negligible in λ.

Case 2: $\epsilon \geq \mathsf{negl}(\lambda)$ - i.e noticeable

Therefore now, in Hyb_2, since each look-ahead thread is identical to the execution in the "Abort Check" step, in each look-ahead thread, the probability with which the thread is "Good" is same as ϵ. Therefore, each look-ahead thread, with noticeable probability, is a "Good" thread. Therefore, even if we run 5 look-ahead threads, they are all "Good" with noticeable probability.

Now, let's show that the algorithm $\mathsf{Ext}_{\mathsf{R.ECom}}$ successfully extracts with noticeable probability. Notice that since the invariant holds in Hyb_2, from the soundness of the schemes $\mathsf{WI}, \mathsf{RWI}$ and WZK, in each "Good" thread, for every malicious party P_j and honest party P_i, either the first or second statements must hold true for the proofs given in round 3. That is, for every malicious party P_j and honest party P_i, either the values $(\mathsf{r.ec}_{a,1}^{j\to i}, \mathsf{r.ec}_{a,2}^{i\to j}, \mathsf{r.ec}_{a,3}^{j\to i})$ or the values $(\mathsf{r.ec}_{b,1}^{j\to i}, \mathsf{r.ec}_{b,2}^{i\to j}, \mathsf{r.ec}_{b,3}^{j\to i})$ form a "well-formed" tuple of the scheme $\mathsf{R.ECom}$ as defined in Section 4.4.

By the definition of $\mathsf{R.ECom}$, algorithm $\mathsf{Ext}_{\mathsf{R.ECom}}$ is successful except with negligible probability if given as input $(\mathsf{r.ec}_1, \{\mathsf{r.ec}_2^k, \mathsf{r.ec}_3^k\}_{k=1}^5)$ such that $(\mathsf{r.ec}_1, \mathsf{r.ec}_2^k, \mathsf{r.ec}_3^k)$ constitute "well-formed" extractable commitment messages for all k. Since the total number of "Good" look-ahead threads is at least 5 with noticeable probability and all the extractable commitments are "well-formed", $\mathsf{Ext}_{\mathsf{R.ECom}}$ extracts successfully with noticeable probability.

By the definition of the scheme TDGen, the algorithm TDExt is successful except with negligible probability if given as input $(\mathsf{td}_1, \{\mathsf{td}_2^i, \mathsf{td}_3^i\}_{i=1}^3)$ where td_1 is the first message of the protocol TDGen and $\mathsf{td}_2^i, \mathsf{td}_3^i$ denote the second and round messages of the i^{th} execution of protocol TDGen using the same first round message. Since the number of "Good" threads is larger than 2 with noticeable probability, the extraction is successful with noticeable probability.

Since both TDExt and Ext are successful with noticeable probability, $\mathsf{Pr}[\ \mathsf{Sim}_{\mathsf{Hyb}}$ outputs "Special Abort" $\ ] \leq \mathsf{negl}(\lambda)$ in Hyb_2 and this completes the proof.

Remark: We will use the same argument in every subsequent hybrid when we argue that the number of "Good" look-ahead threads is enough for the reduction to extract with noticeable probability. □

Claim 13. *If the "Check Abort" step succeeds, assuming* NMCom *is a secure non-malleable commitment scheme, the invariant holds in* Hyb_3.

Proof. We know that the invariant holds in Hyb_2. The only difference between Hyb_2 and Hyb_3 is that in Hyb_3, the simulator now computes the non-malleable commitment in the main thread using the adversary's trapdoor value. Assume for the sake of contradiction that the invariant doesn't hold in Hyb_3. That is, there exists an adversary $\mathcal{A}$ such that for some honest party P_i^* and malicious party P_j^*, $\mathcal{A}$ causes event E to occur in the main thread with non-negligible probability. We will use $\mathcal{A}$ to design an adversary $\mathcal{A}_{\mathsf{NMCom}}$ that breaks the security of the non-malleable commitment scheme.

$\mathcal{A}_{\mathsf{NMCom}}$ interacts with a challenger $\mathcal{C}_{\mathsf{NMCom}}$. $\mathcal{A}_{\mathsf{NMCom}}$ performs the role of $\mathsf{Sim}_{\mathsf{Hyb}}$ in its interaction with $\mathcal{A}$ exactly as done in Hyb_2. $\mathcal{A}_{\mathsf{NMCom}}$ picks an honest party P_i and a malicious party P_j uniformly at random. $\mathcal{A}_{\mathsf{NMCom}}$ interacts with a challenger $\mathcal{C}_{\mathsf{NMCom}}$ and receives a first round message c_1^L on the left side which is set as $\mathsf{c}_1^{i\to j}$ in its interaction with $\mathcal{A}$ in round 1 of protocol π. On receiving $\mathsf{c}_1^{j\to i}$, $\mathcal{A}_{\mathsf{NMCom}}$ forwards this to $\mathcal{C}_{\mathsf{NMCom}}$ as its first round message on the right side.

$\mathcal{A}_{\mathsf{NMCom}}$ creates a set of 5 look-ahead threads, in each of which, it runs rounds 2 and 3 of the protocol alone. In each look-ahead thread, $\mathcal{A}_{\mathsf{NMCom}}$ computes $\mathsf{c}_3^{i\to j}$ as a commitment to $\bot$. Recall from the definition of the scheme NMCom that $\mathcal{A}_{\mathsf{NMCom}}$ can do this even without knowing the randomness used to generate $\mathsf{c}_1^{i\to j}$. As in the proof of Claim 12, recall that using these 5 threads, $\mathcal{A}_{\mathsf{NMCom}}$ can extract with noticeable probability and thus, successfully run the input extraction phase.

Then, on the main thread, $\mathcal{A}_{\mathsf{NMCom}}$ receives a value c_2^R from $\mathcal{C}_{\mathsf{NMCom}}$ as the second round message on the right side which it sets as the value $\mathsf{c}_2^{i\to j}$ in round 2 of protocol π on the main thread. Then, on receiving $\mathsf{c}_2^{j\to i}$ in the main thread, $\mathcal{A}_{\mathsf{NMCom}}$ sends this to $\mathcal{C}_{\mathsf{NMCom}}$ as its second round message on the left side along with the pair of values $(\bot, \mathsf{t}_j)$ where t_j is generated in the input extraction phase. Recall that NMCom is a delayed-input scheme.

$\mathcal{A}_{\mathsf{NMCom}}$ receives a third round message c_3^L which is either a commitment to $\bot$ or t_j. This is sent to $\mathcal{A}$ as the value $\mathsf{c}_3^{i\to j}$ in the main thread and the rest of protocol π is performed exactly as in Hyb_1. On receiving the value $\mathsf{c}_3^{i\to j}$ from $\mathcal{A}$ in the main thread, $\mathcal{A}_{\mathsf{NMCom}}$ sends it to $\mathcal{C}_{\mathsf{NMCom}}$ as its third round message in the right thread. Note that in all the other look-ahead threads, $\mathcal{A}_{\mathsf{NMCom}}$ continues to compute the non-malleable commitment using input $\bot$ as done in Hyb_1. Recall that $\mathsf{NMCom}_3(\bot)$ is indistinguishable from a random string and so $\mathcal{A}_{\mathsf{NMCom}}$ can generate this by picking a uniformly random string without knowing the randomness used to generate $\mathsf{c}_1^{i\to j}$.

Observe that the first case corresponds to Hyb_2 while the second case corresponds to Hyb_3. Now, let's analyze why $\mathcal{A}_{\mathsf{NMCom}}$ breaks the security of the scheme NMCom. We know that the invariant doesn't hold in Hyb_3 so there exists honest party P_i^* and malicious party P_j^* such that event E doesn't hold. That is, the adversary P_j^*, using the non-malleable commitment, commits to a valid trapdoor t_i^* for the trapdoor generation messages of the honest party P_i^* with non-negligible probability ϵ. With probability $\frac{1}{n^2}$ where n is the total number of parties, this corresponds to honest party P_i and malicious party P_j picked randomly by $\mathcal{A}_{\mathsf{NMCom}}$. Therefore, with non-negligible probability $\frac{\epsilon}{n^2}$, the adversary P_j, using the non-malleable commitment, commits to a valid trapdoor t_i^* for the trapdoor generation messages of the honest party P_i in Hyb_3. Recall that this is same as the messages sent by $\mathcal{A}_{\mathsf{NMCom}}$ to $\mathcal{C}_{\mathsf{NMCom}}$ as its commitment messages on the right side. In Hyb_2, since

the invariant holds, the adversary did not commit to a valid trapdoor from any malicious party P_j to honest party P_i.

Therefore, when the value committed to by the honest party in the left execution changes, the value committed to by the adversary in the right execution has also changed except with negligible probability. This breaks the security of the scheme NMCom which is a contradiction. Thus, the invariant must hold in Hyb_2 as well. □

Claim 14. *Assuming the hiding property of the non-malleable commitment scheme* NMCom, Hyb_2 *is computationally indistinguishable from* Hyb_3.

Proof. The only difference between Hyb_2 and Hyb_3 is that in Hyb_3, the simulator now computes the non-malleable commitment using the adversary's trapdoor value. Suppose there exists an adversary $\mathcal{A}$ that can distinguish between the two hybrids. We can use $\mathcal{A}$ to design an adversary $\mathcal{A}_{\mathsf{Hid}}$ that breaks the hiding of the non-malleable commitment scheme. The rest of the proof is similar to the proof of Claim 13 above.

□

Claim 15. *If the "Check Abort" step succeeds, Assuming the bounded rewinding security of the protocol* RWI *and the existence of an extractor* $\mathsf{Ext}_{\mathsf{NMCom}}$ *for the non-malleable commitment scheme* NMCom, *the invariant holds in* Hyb_4.

Proof. We know that the invariant holds in Hyb_3. The only difference between Hyb_3 and Hyb_4 is that in Hyb_4, in the main thread, the simulator now computes the rewinding secure WI using a witness for the non-malleable commitment. Assume for the sake of contradiction that the invariant doesn't hold in Hyb_4. That is, there exists an adversary $\mathcal{A}$ such that for some honest party P_i^* and malicious party P_j^*, $\mathcal{A}$ causes event E to occur in the main thread with non-negligible probability.

We will use $\mathcal{A}$ to design an adversary $\mathcal{A}_{\mathsf{RWI}}$ that breaks the bounded security of the protocol RWI.

$\mathcal{A}_{\mathsf{RWI}}$ interacts with a challenger $\mathcal{C}_{\mathsf{RWI}}$. $\mathcal{A}_{\mathsf{RWI}}$ performs the role of $\mathsf{Sim}_{\mathsf{Hyb}}$ in its interaction with $\mathcal{A}$ exactly as done in Hyb_2. $\mathcal{A}_{\mathsf{RWI}}$ picks an honest party P_i and a malicious party P_j uniformly at random. Initially, it receives a message rwi_1 from $\mathcal{C}_{\mathsf{RWI}}$ which it sets as the value $\mathsf{rwi}_1^{i\to j}$ in its interaction with $\mathcal{A}$ in round 1. Then, $\mathcal{A}_{\mathsf{RWI}}$ creates a set of 5 look-ahead threads. For each thread, on receiving $\mathsf{rwi}_2^{j\to i}$ in round 2, $\mathcal{A}_{\mathsf{RWI}}$ forwards this to $\mathcal{C}_{\mathsf{RWI}}$ as its second round message for that look-ahead thread. For each thread, $\mathcal{A}_{\mathsf{RWI}}$ also sends the statement $\mathsf{st}_2^{i\to j} = (\mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{a,2}^{j\to i}, \mathsf{r.ec}_{a,3}^{i\to j}, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}, \mathsf{r.ec}_{b,3}^{i\to j}, \mathsf{msg}_{2,i}, \mathsf{Trans}_1, \mathsf{c}_1^{i\to j}, \mathsf{c}_2^{j\to i}, \mathsf{c}_3^{i\to j}, \mathsf{td}_1^{j\to i}, \mathsf{nc}_i) \in L_2^{i\to j}$ where the other values are generated as in Hyb_3.

In the main thread, $\mathcal{A}_{\mathsf{RWI}}$ also sends the pair of witnesses $(\mathsf{x}_i, \mathsf{r}_i, \mathsf{r}_{\mathsf{r.ec}}^{i\to j}, \bot, \bot, \bot, \bot)$ and $(\bot, \bot, \bot, \bot, \mathsf{t}_j, \mathsf{r}_{\mathsf{c}}^{i\to j}, \bot)$ where t_j is generated in the input extraction phase. For each look-ahead thread, $\mathcal{A}_{\mathsf{RWI}}$ sends the witness $(\mathsf{x}_i, \mathsf{r}_i, \mathsf{r}_{\mathsf{r.ec}}^{i\to j}, \bot, \bot, \bot, \bot)$. For each thread, $\mathcal{A}_{\mathsf{RWI}}$ receives a message rwi_3 which is set as $\mathsf{rwi}_3^{i\to j}$ in its interaction with $\mathcal{A}$ in round 3 of protocol π. The rest of protocol π is performed exactly as in Hyb_3. In particular, recall from the proof of Claim 12 that each look-ahead thread is "Good" with noticeable probability and hence $\mathcal{A}_{\mathsf{RWI}}$ can successfully run the input extraction phase.

Observe that the first case corresponds to Hyb_3 while the second case corresponds to Hyb_4.

Now, let's see how $\mathcal{A}_{\mathsf{RWI}}$ breaks the security of the protocol RWI. Recall that RWI is secure even in the presence of 6 rewound look-ahead threads (we set L to be 6). $\mathcal{A}_{\mathsf{RWI}}$ runs the extractor $\mathsf{Ext}_{\mathsf{NMCom}}$ of the non-malleable commitment scheme using the messages in all the threads that correspond to the non-malleable commitment from malicious party P_j to honest party P_i. Let the output of $\mathsf{Ext}_{\mathsf{NMCom}}$ be t^*. If $\mathsf{TDValid}(\mathsf{td}_1^{i\to j}, \mathsf{t}^*) = 1$, then $\mathcal{A}_{\mathsf{RWI}}$ outputs case 2 indicating that the WI was constructed using the second witness on the main thread. Else, it outputs case 1.

Let's analyze why this works. We know that the invariant doesn't hold in Hyb_4 so there exists honest party P_i^* and malicious party P_j^* such that event E doesn't hold. That is, the adversary P_j^*, using the non-malleable commitment, commits to a valid trapdoor t_i^* for the trapdoor generation messages of the honest party P_i^* with non-negligible probability ϵ. With probability $\frac{1}{n^2}$ where n is the total number of parties, this corresponds to honest party P_i and malicious party P_j picked randomly by $\mathcal{A}_{\mathsf{RWI}}$. Therefore, with non-negligible probability $\frac{\epsilon}{n^2}$, the adversary P_j, using the non-malleable commitment, commits to a valid trapdoor t_i^* for the trapdoor generation messages of the honest party P_i in Hyb_3. Therefore, since the invariant holds in Hyb_3 with non-negligible probability, when the extractor $\mathsf{Ext}_{\mathsf{NMCom}}$ outputs a valid trapdoor t^*, it must be the case that we are in Hyb_4 with non-negligible probability. That is, when $\mathsf{Ext}_{\mathsf{NMCom}}$ outputs a valid trapdoor, it corresponds to $\mathcal{A}_{\mathsf{RWI}}$ receiving a proof using the second (alternate) witness and otherwise, it corresponds to $\mathcal{A}_{\mathsf{RWI}}$ receiving a proof using the first witness. Thus, $\mathcal{A}_{\mathsf{RWI}}$ breaks the bounded rewinding security of the scheme RWI which is a contradiction. Hence, the invariant holds in Hyb_4 as well.

□

Claim 16. *Assuming the bounded rewinding security of the protocol* RWI*,* Hyb_3 *is indistinguishable from* Hyb_4*.*

Proof. The only difference between Hyb_3 and Hyb_4 is that in Hyb_4, in the main thread, the simulator now computes the rewinding secure WI using a witness for the alternate statement. Suppose there exists an adversary $\mathcal{A}$ that can distinguish between the two hybrids, we can use $\mathcal{A}$ to design an adversary $\mathcal{A}_{\mathsf{RWI}}$ that breaks the bounded rewinding security of the protocol RWI. The rest of the proof is very similar to the proof of Claim 15 above.

□

Claim 17. *If the "Check Abort" step succeeds, assuming* WI *is a secure delayed input witness indistinguishable argument, the invariant holds in* Hyb_5*.*

Proof. Notice that the only difference between Hyb_4 and Hyb_5 is in the witness used in the WI proof of the main thread that is completed in round 4 of the protocol. However, since WI is a delayed-input scheme, there is no difference in the first 2 rounds. In other words, there is no difference between Hyb_4 and Hyb_5 in the messages used in the first 3 rounds of the protocol. Since the invariant depends only on the first 3 rounds, and since the invariant holds in Hyb_4, it continues to hold in Hyb_5. □

Claim 18. *Assuming* WI *is a secure delayed input witness indistinguishable argument,* Hyb_4 *is indistinguishable from* Hyb_5*.*

Proof. This is very similar to the proof of Claim 16 with the only difference being the reduction uses the external challenger's messages only once - to generate the proof in the final round of the main thread and not in each look-ahead thread. Therefore, we don't need any rewinding security for the scheme WI. □

Claim 19. *If the "Check Abort" step succeeds, assuming the bounded rewinding security of the* RWI *, the hiding property of the underlying commitment scheme* Com *used in the construction of* $\mathsf{R.ECom}$ *and the existence of an extractor* $\mathsf{Ext}_{\mathsf{NMCom}}$ *for the non-malleable commitment scheme* NMCom*, the invariant holds in* Hyb_6*.*

Proof. We know that the invariant holds in Hyb_5. The only difference between Hyb_5 and Hyb_6 is that in Hyb_6, the simulator now computes the extractable commitment in the main thread for every honest party P_i using input 0 whereas in Hyb_5 it was computed using input $(\mathsf{x}_i, \mathsf{r}_i)$.

First, throughout this proof, we drop the subscript "a" since it is clear from context that we are referring only to the first invocation of the extractable commitment. The second invocation is not used in the proof at all.

Let's briefly recall the construction of the scheme $\mathsf{R.ECom}$ from Section 4.4 in the context of protocol π. Consider a party P_i as committer interacting with a party P_j as receiver using input message m. In round 1, P_i computes $\mathsf{r.ec}_1^{i\to j} = \mathsf{brew.ec}_1^{i\to j}$ where $\mathsf{brew.ec}_1^{i\to j} = \mathsf{BRew.ECom}_1(\mathsf{r}^{i\to j}; \mathsf{r}_{\mathsf{brew.ec}})$ using random strings $\mathsf{r}^{i\to j}, \mathsf{r}_{\mathsf{brew.ec}}$. In round 2, P_j responds with $\mathsf{r.ec}_2^{j\to i} = \mathsf{brew.ec}_2^{j\to i}$ where $\mathsf{brew.ec}_2^{j\to i} = \mathsf{BRew.ECom}_2(\mathsf{brew.ec}_1^{i\to j})$. Finally, in round 3, P_i computes $\mathsf{r.ec}_3^{i\to j}$ as $(\alpha^{i\to j}, \mathsf{brew.ec}_3^{i\to j}, \beta^{i\to j})$ where $\mathsf{brew.ec}_3^{i\to j} = \mathsf{BRew.ECom}_3(\mathsf{r}_{i,j}, \mathsf{brew.ec}_1^{i\to j}, \mathsf{brew.ec}_2^{j\to i}; \mathsf{r}_{\mathsf{brew.ec}})$, $\beta_b^{i\to j} = \mathsf{PRF}(\mathsf{r}^{i\to j}, \alpha^{i\to j}) \oplus m$.

Now, Let's zoom in and recall the construction of the scheme $\mathsf{BRew.ECom}$ from Section 4.3 in the context of protocol π. In round 1, P_i computes $\mathsf{brew.ec}_1^{i\to j} = \{\mathsf{brew.ec}_{1,1}^{i\to j}, \ldots, \mathsf{brew.ec}_{1,\lambda}^{i\to j}\}$ where $\mathsf{brew.ec}_{1,l}^{i\to j} = \mathsf{Com}(\mathsf{p}_l^{i\to j})$ where each $\mathsf{p}_l^{i\to j}$ is a random polynomial of degree 4 (defined in Section 4.3). In round 2, P_j generates $\mathsf{brew.ec}_2^{j\to i} = (\mathsf{z}_1^{j\to i}, \ldots, \mathsf{z}_\lambda^{j\to i})$ where each $\mathsf{z}_l^{j\to i}$ is a random value. Then, in round 3, P_i outputs $\mathsf{brew.ec}_3^{i\to j} = \{\mathsf{brew.ec}_{3,1}^{i\to j}, \ldots, \mathsf{brew.ec}_{3,\lambda}^{i\to j}\}$ where $\mathsf{brew.ec}_{3,l}^{i\to j} = (\mathsf{r}^{i\to j} \oplus \mathsf{p}_l^{i\to j}(0), \mathsf{p}_l^{i\to j}(\mathsf{z}_l^{j\to i}))$ for each $l \in [\lambda]$.

We will design a set of intermediate hybrids $\mathsf{Sub.Hyb}_1$ to $\mathsf{Sub.Hyb}_5$ where $\mathsf{Sub.Hyb}_1$ denotes Hyb_5 and $\mathsf{Sub.Hyb}_5$ denotes Hyb_6. We will then show that the invariant holds in every intermediate hybrid to complete the proof.

- $\mathsf{Sub.Hyb}_1$**:** This is same as H_5.
- $\mathsf{Sub.Hyb}_2$**:** In this hybrid, for every honest party P_i and malicious party P_j, only on the main thread, compute $\mathsf{brew.ec}_3^{i\to j} = \mathsf{BRew.ECom}_3(0, \mathsf{brew.ec}_1^{i\to j}, \mathsf{brew.ec}_2^{j\to i}; \mathsf{r}_{\mathsf{brew.ec}})$. Observe that all the look-ahead threads continue committing to a random value $\mathsf{r}^{i\to j}$.
- $\mathsf{Sub.Hyb}_3$**:** In this hybrid, for every honest party P_i and malicious party P_j, in the main thread, compute $\beta^{i\to j}$ uniformly at random.
- $\mathsf{Sub.Hyb}_4$**:** In this hybrid, for every honest party P_i and malicious party P_j, in the main thread, compute $\beta^{i\to j} = \mathsf{PRF}(\mathsf{r}^{i\to j}, \alpha^{i\to j}) \oplus 0$.
- $\mathsf{Sub.Hyb}_5$**:** In this hybrid, for every honest party P_i and malicious party P_j, compute $\mathsf{brew.ec}_3^{i\to j} = \mathsf{BRew.ECom}_3(\mathsf{r}^{i\to j}, \mathsf{brew.ec}_1^{i\to j}, \mathsf{brew.ec}_2^{j\to i}; \mathsf{r}_{\mathsf{brew.ec}})$. This is same as H_6.

Sub-Claim 1. *Assuming the bounded rewinding security of the* RWI*, the hiding property of the underlying commitment scheme* Com *used in the construction of* $\mathsf{BRew.ECom}$ *and the existence of an extractor* $\mathsf{Ext}_{\mathsf{NMCom}}$ *for the non-malleable commitment scheme* NMCom*, the invariant holds in* $\mathsf{Sub.Hyb}_2$*.*

Proof. The only difference between the two hybrids is that in $\mathsf{Sub.Hyb}_1$, the underlying commitment $\mathsf{BRew.ECom}$ is to $\mathsf{r}^{i\to j}$ while in $\mathsf{Sub.Hyb}_2$, it is to 0. We know that the invariant holds in $\mathsf{Sub.Hyb}_1$.

We will design a set of intermediate hybrids $\mathsf{Sub.Hyb}_{1,0}$ to $\mathsf{Sub.Hyb}_{1,\lambda}$ where $\mathsf{Sub.Hyb}_{1,0}$ denotes $\mathsf{Sub.Hyb}_1$ and $\mathsf{Sub.Hyb}_{1,\lambda}$ denotes $\mathsf{Sub.Hyb}_2$. We will then show that the invariant holds in every intermediate hybrid to complete the proof. Here, λ is the security parameter. **For** $l = 1 \ldots \lambda$**,**

$\mathsf{Sub.Hyb}_{1,l}$**:** For every honest party P_i and malicious party P_j, do: (to ease the exposition, we will skip the superscript $i \to j$).

- Pick a new degree 4 polynomial q_l such that $(\mathsf{r} \oplus \mathsf{p}_l(0)) = (0 \oplus \mathsf{q}_l(0))$.
- In round 1, compute $\mathsf{brew.ec}_{1,l} = \mathsf{Com}(\mathsf{q}_l)$.
- in the main thread, compute $\mathsf{brew.ec}_{3,l}$ as $(0 \oplus \mathsf{q}_l(0), \mathsf{q}_l(\mathsf{z}_l))$.
- In every look-ahead thread, compute $\mathsf{brew.ec}_{3,l}$ as before, i.e using input r but using polynomial q_l.

That is, in $\mathsf{Sub.Hyb}_{1,l}$, we switch the l^{th} index of the extractable commitment scheme from using input (r) to using input 0.

Recall that the goal was to show that the invariant holds in every intermediate hybrid. Assume for the sake of contradiction that there exists l such that the invariant doesn't hold in $\mathsf{Sub.Hyb}_{1,l}$ but holds in $\mathsf{Sub.Hyb}_{1,0}, \ldots, \mathsf{Sub.Hyb}_{1,l-1}$. That is, there exists an adversary $\mathcal{A}$ such that for some honest party P_i^* and malicious party P_j^*, $\mathcal{A}$ causes event E to occur with non-negligible probability in $\mathsf{Sub.Hyb}_{1,l}$. We will use $\mathcal{A}$ to arrive at a contradiction.

We will again prove this using a series of intermediate hybrids. We know that the invariant holds in $\mathsf{Sub.Hyb}_{1,l-1}$. Consider a set of hybrids $\mathsf{H}_1, \ldots, \mathsf{H}_5$ as follows where H_1 corresponds to $\mathsf{Sub.Hyb}_{1,l-1}$ and H_5 corresponds to $\mathsf{Sub.Hyb}_{1,l}$.

- H_2**:** In round 3 of every look-ahead thread, for every honest party P_i and malicious party P_j, the algorithm RWI_3 proves that the commitment $(\mathsf{brew.ec}_1, \mathsf{brew.ec}_2, \mathsf{brew.ec}_3)$ is "well-formed" using all indices from $1, \ldots, \lambda$ except index l. Recall that proving well-formedness of the commitment using the scheme $\mathsf{R.ECom}$ essentially boils down to proving well-formedness of the underlying commitment using the scheme $\mathsf{BRew.ECom}$.
- H_3**:** In round 1, for every honest party P_i and malicious party P_j, compute $\mathsf{brew.ec}_{1,l} = \mathsf{Com}(0)$.
- H_4**:** For every honest party P_i and malicious party P_j:
  - In the main thread, compute $\mathsf{brew.ec}_{3,l}$ as $(0 \oplus \mathsf{q}_l(0), \mathsf{q}_l(\mathsf{z}_l))$.
  - In every look-ahead thread, compute $\mathsf{brew.ec}_{3,l}$ as before but using polynomial q_l.
- H_5**:** In round 1, for every honest party P_i and malicious party P_j, compute $\mathsf{brew.ec}_{1,l} = \mathsf{Com}(\mathsf{q}_l)$.

Note: as before, we drop the superscript $i \to j$ to ease the exposition. We will now show that the invariant holds in $\mathsf{H}_2, \ldots, \mathsf{H}_5$ and this completes the proof.

Lemma 1. *Invariant holds in* H_2.

Proof. We know that the invariant holds in H_1. Suppose it doesn't hold in H_2. That is, there exists an adversary $\mathcal{A}$ such that for some honest party P_i^* and malicious party P_j^*, $\mathcal{A}$ causes event E to occur with non-negligible probability. The only difference between the two hybrids is in the witness used to compute the WI in round 3 of protocol π for every look-ahead thread. We will use $\mathcal{A}$ to design an adversary $\mathcal{A}_{\mathsf{RWI}}$ that breaks the bounded rewinding security of the scheme RWI.

$\mathcal{A}_{\mathsf{RWI}}$ interacts with a challenger $\mathcal{C}_{\mathsf{RWI}}$. $\mathcal{A}_{\mathsf{RWI}}$ performs the role of $\mathsf{Sim}_{\mathsf{Hyb}}$ in its interaction with $\mathcal{A}$ exactly as done in H_1. $\mathcal{A}_{\mathsf{RWI}}$ picks an honest party P_i, a malicious party P_j. Initially, it receives a

message rwi_1 from $\mathcal{C}_{\mathsf{RWI}}$ which it sets as the value $\mathsf{rwi}_1^{i\to j}$ in its interaction with $\mathcal{A}$ in round 1. Then, $\mathcal{A}_{\mathsf{RWI}}$ creates a set of 5 look-ahead threads. On receiving $\mathsf{rwi}_2^{j\to i}$ in round 2 for each look-ahead thread, $\mathcal{A}_{\mathsf{RWI}}$ forwards this to $\mathcal{C}_{\mathsf{RWI}}$ as its second round message for that look-ahead thread. $\mathcal{A}_{\mathsf{RWI}}$ also sends the statement $\mathsf{st}_2^{i\to j}$ =($\mathsf{brew.ec}_1^{i\to j}, \mathsf{brew.ec}_2^{j\to i}, \mathsf{brew.ec}_3^{i\to j}, \mathsf{msg}_{2,i}, \mathsf{Trans}_1, \mathsf{c}_1^{i\to j}, \mathsf{c}_2^{j\to i}, \mathsf{c}_3^{i\to j}, \mathsf{td}_1^{j\to i}, \mathsf{td}_2^{i\to j}, \mathsf{td}_3^{j\to i}$) $\in L_2^{i\to j}$ where the other values are generated as in H_1 along with the pair of witnesses used in H_1 and H_2 respectively. Corresponding to each look-ahead thread, $\mathcal{A}_{\mathsf{RWI}}$ receives a message rwi_3 which is set as $\mathsf{rwi}_3^{i\to j}$ in its interaction with $\mathcal{A}$ in round 3 of protocol π. The rest of protocol π is performed exactly as in H_1. In particular, recall from the proof of Claim 12 that each look-ahead thread is "Good" with noticeable probability and hence $\mathcal{A}_{\mathsf{RWI}}$ can successfully run the input extraction phase.

Observe that the first case corresponds to H_1 while the second case corresponds to H_2.

Now, let's see how $\mathcal{A}_{\mathsf{RWI}}$ breaks the bounded rewinding security of the scheme RWI. $\mathcal{A}_{\mathsf{RWI}}$ runs the extractor $\mathsf{Ext}_{\mathsf{NMCom}}$ of the non-malleable commitment scheme using the messages in all the threads that correspond to the non-malleable commitment from malicious party P_j to honest party P_i. Let the output of $\mathsf{Ext}_{\mathsf{NMCom}}$ be t^*. If $\mathsf{TDValid}(\mathsf{td}_1^{i\to j}, \mathsf{td}_2^{j\to i}, \mathsf{td}_3^{i\to j}, \mathsf{t}^*) = 1$, then $\mathcal{A}_{\mathsf{RWI}}$ outputs case 2 indicating that the WI was constructed using the witness as in H_2. Else, it outputs case 1.

Let's analyze why this works. We know that the invariant doesn't hold in H_2 so there exists honest party P_i^* and malicious party P_j^* such that event E doesn't hold. That is, the adversary P_j^*, using the non-malleable commitment, commits to a valid trapdoor t_i^* for the trapdoor generation messages of the honest party P_i^* with non-negligible probability ϵ. With probability $\frac{1}{n^2}$ where n is the total number of parties, this corresponds to honest party P_i and malicious party P_j picked randomly by $\mathcal{A}_{\mathsf{RWI}}$. Therefore, with non-negligible probability $\frac{\epsilon}{n^2}$, the adversary P_j, using the non-malleable commitment, commits to a valid trapdoor t_i^* for the trapdoor generation messages of the honest party P_i in H_2. Therefore, since the invariant holds in H_1 with non-negligible probability, when the extractor $\mathsf{Ext}_{\mathsf{NMCom}}$ outputs a valid trapdoor t^*, it must be the case that we are in H_2 with non-negligible probability. Thus, $\mathcal{A}_{\mathsf{RWI}}$ breaks the bounded rewinding security of the scheme RWI which is a contradiction. Hence, the invariant holds in H_2 as well. □

Lemma 2. *Invariant holds in* H_3.

Proof. We know that the invariant holds in H_2. The only difference between H_2 and H_3 is that in H_3, the simulator now computes the commitment $\mathsf{brew.ec}_{1,l}$ in round 1 as $\mathsf{brew.ec}_{1,l} = \mathsf{Com}(0)$ for every honest party P_i and malicious party P_j. Assume for the sake of contradiction that the invariant doesn't hold in H_3. That is, there exists an adversary $\mathcal{A}$ such that for some honest party P_i^* and malicious party P_j^*, $\mathcal{A}$ causes event E to occur with non-negligible probability. We will use $\mathcal{A}$ to design an adversary $\mathcal{A}_{\mathsf{Com}}$ that breaks the hiding of the commitment scheme.

$\mathcal{A}_{\mathsf{Com}}$ interacts with a challenger $\mathcal{C}_{\mathsf{Com}}$. $\mathcal{A}_{\mathsf{Com}}$ performs the role of $\mathsf{Sim}_{\mathsf{Hyb}}$ in its interaction with $\mathcal{A}$ exactly as done in Hyb_1. $\mathcal{A}_{\mathsf{NMCom}}$ picks an honest party P_i and a malicious party P_j uniformly at random. $\mathcal{A}_{\mathsf{Com}}$ creates a set of 5 look-ahead threads. $\mathcal{A}_{\mathsf{Com}}$ interacts with a challenger $\mathcal{C}_{\mathsf{Com}}$ and sends two strings: (p_l) and (0). $\mathcal{A}_{\mathsf{Com}}$ receives a message c which is sent to $\mathcal{A}$ as the value $\mathsf{brew.ec}_{1,l}$ round 1 and the rest of protocol π is performed exactly as in H_2. In particular, recall from the proof of Claim 12 that each look-ahead thread is "Good" with noticeable probability and hence $\mathcal{A}_{\mathsf{Com}}$ can successfully run the input extraction phase.

Observe that the first case corresponds to H_2 while the second case corresponds to H_3.

Now, let's analyze why $\mathcal{A}_{\mathsf{Com}}$ breaks the hiding of the scheme Com. We know that the invariant doesn't hold in H_3 so there exists honest party P_i^* and malicious party P_j^* such that event E doesn't hold. That is, the adversary P_j^*, using the non-malleable commitment, commits to a valid trapdoor t_i^* for the trapdoor generation messages of the honest party P_i^* with non-negligible probability ϵ.

With probability $\frac{1}{n^2}$ where n is the total number of parties, this corresponds to honest party P_i and malicious party P_j picked randomly by $\mathcal{A}_{\mathsf{Com}}$. Therefore, with non-negligible probability $\frac{\epsilon}{n^2}$, the adversary P_j, using the non-malleable commitment, commits to a valid trapdoor t_i^* for the trapdoor generation messages of the honest party P_i in H_3. Therefore, since the invariant holds in H_2 with non-negligible probability, when the extractor $\mathsf{Ext}_{\mathsf{NMCom}}$ outputs a valid trapdoor t^*, it must be the case that we are in H_3 with non-negligible probability. Thus, $\mathcal{A}_{\mathsf{Com}}$ can use this to break the hiding of the commitment scheme Com which is a contradiction. Hence, the invariant holds in H_3 as well. □

Lemma 3. *Invariant holds in* H_4.

Proof. We know that the invariant holds in H_3. The difference between H_3 and H_4 is only a statistical change and hence the invariant continues to hold in H_4 as well. □

Lemma 4. *Invariant holds in* H_5.

Proof. This is same as the proof of Lemma 2. □

This completes the proof of Sub-Claim 1. □

Sub-Claim 2. *Assuming the security of the pseudorandom function* PRF *and the existence of an extractor* $\mathsf{Ext}_{\mathsf{NMCom}}$ *for the non-malleable commitment scheme* NMCom*, the invariant holds in* $\mathsf{Sub.Hyb}_3$.

Proof. The only difference between the two hybrids is that, in $\mathsf{Sub.Hyb}_3$, on the main thread, the value $\beta^{i\to j}$ is computed uniformly at random while in $\mathsf{Sub.Hyb}_2$, it was computed as $\mathsf{PRF}(\mathsf{r}^{i\to j}, \alpha^{i\to j}) \oplus \mathsf{x}_i$. Therefore, if there exists an adversary $\mathcal{A}$ for which the invariant holds in $\mathsf{Sub.Hyb}_2$ but not in $\mathsf{Sub.Hyb}_3$, we can use the extractor $\mathsf{Ext}_{\mathsf{NMCom}}$ to extract the value inside the adversary's non-malleable commitment and check whether the invariant holds or not. Thus, we can build a reduction that breaks the security of the PRF. □

Sub-Claim 3. *Assuming the security of the pseudorandom function* PRF *and the existence of an extractor* $\mathsf{Ext}_{\mathsf{NMCom}}$ *for the non-malleable commitment scheme* NMCom*, the invariant holds in* $\mathsf{Sub.Hyb}_4$.

Proof. This is identical to the previous proof. □

Sub-Claim 4. *Assuming the bounded rewinding security of the scheme* RWI*, the hiding property of the underlying commitment scheme* Com *used in the construction of* BRew.ECom *and the existence of an extractor* $\mathsf{Ext}_{\mathsf{NMCom}}$ *for the non-malleable commitment scheme* NMCom*, the invariant holds in* $\mathsf{Sub.Hyb}_5$.

Proof. This is identical to the proof of Sub-Claim 1. □

This completes the proof of Claim 19. □

Claim 20. *Assuming the bounded rewinding security of the scheme* RWI *and the hiding property of the underlying commitment scheme* Com *used in the construction of* R.ECom*,* Hyb_5 *is indistinguishable from* Hyb_6.

Proof. This is similar to the proof of Claim 19.

□

Claim 21. *If the "Check Abort" step succeeds, assuming the security of protocol* π^{SM} *and the existence of an extractor* $\mathsf{Ext}_{\mathsf{NMCom}}$ *for the non-malleable commitment scheme* NMCom*, the invariant holds in* Hyb_7*.*

Proof. We know that the invariant holds in Hyb_6. The only difference between Hyb_6 and Hyb_7 is that in Hyb_7, in the main thread, the simulator now computes the messages of protocol π^{SM} using the simulated algorithms $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$. Assume for the sake of contradiction that the invariant doesn't hold in Hyb_6. That is, there exists an adversary $\mathcal{A}$ such that for some honest party P_i^* and malicious party P_j^*, $\mathcal{A}$ causes event E to occur in the main thread with non-negligible probability. We will use $\mathcal{A}$ to design an adversary $\mathcal{A}_{\pi^{\mathsf{SM}}}$ that breaks the security of the protocol π^{SM}.

$\mathcal{A}_{\pi^{\mathsf{SM}}}$ performs the role of $\mathsf{Sim}_{\mathsf{Hyb}}$ in its interaction with $\mathcal{A}$ exactly as done in Hyb_6. $\mathcal{A}_{\pi^{\mathsf{SM}}}$ also interacts with a challenger $\mathcal{C}_{\pi^{\mathsf{SM}}}$ and corrupts the same parties as done by $\mathcal{A}$. For every honest party P_i, $\mathcal{A}_{\pi^{\mathsf{SM}}}$ receives a first round message $\mathsf{msg}_{1,i}$ which is sent to $\mathcal{A}$ in round 1 of protocol π on the main thread. On receiving $\mathsf{msg}_{1,j}$ for every malicious party P_j in round 1 of the main thread from $\mathcal{A}$, $\mathcal{A}_{\pi^{\mathsf{SM}}}$ forwards this to $\mathcal{C}_{\pi^{\mathsf{SM}}}$ as the first round messages of the malicious parties.

$\mathcal{A}_{\pi^{\mathsf{SM}}}$ then creates a set of 5 look-ahead threads, in each of which, it runs rounds 2 and 3 of the protocol alone. In each look-ahead thread, $\mathcal{A}_{\mathsf{NMCom}}$ computes msg_2^i using input 0. Recall from the properties of the scheme π^{SM} that $\mathcal{A}_{\pi^{\mathsf{SM}}}$ can do this even without knowing the randomness used to generate msg_1^i. As in the proof of Claim 12, recall that using these 5 threads, $\mathcal{A}_{\pi^{\mathsf{SM}}}$ can extract with noticeable probability and thus, successfully run the input extraction phase.

Then, on the main thread, as before, the messages $\mathsf{msg}_{2,i}$ and $\mathsf{msg}_{2,j}$ corresponding to every honest party P_i and malicious party P_j are sent across between $\mathcal{C}_{\pi^{\mathsf{SM}}}$ and $\mathcal{A}$ via $\mathcal{A}_{\pi^{\mathsf{SM}}}$ in round 3 of protocol π on the main thread. $\mathcal{A}_{\pi^{\mathsf{SM}}}$ also sends the values $(\mathsf{y}, \{\mathsf{x}_j, \mathsf{r}_j\})$ (obtained in the input extraction phase) to $\mathcal{C}_{\pi^{\mathsf{SM}}}$. Then, for every honest party P_i $\mathcal{A}_{\pi^{\mathsf{SM}}}$ receives a third round message $\mathsf{msg}_{3,i}$ which is sent to $\mathcal{A}$ in round 4 of protocol π. On receiving $\mathsf{msg}_{3,j}$ for every malicious party P_j in round 3 from $\mathcal{A}$, $\mathcal{A}_{\pi^{\mathsf{SM}}}$ forwards this to $\mathcal{C}_{\pi^{\mathsf{SM}}}$ as the third round messages of the malicious parties. The rest of protocol π is performed exactly as in Hyb_5. Observe that when $\mathcal{C}_{\pi^{\mathsf{SM}}}$ computes the messages of protocol π^{SM} honestly, $\mathcal{A}$'s view corresponds to Hyb_5 and then $\mathcal{C}_{\pi^{\mathsf{SM}}}$ computes simulated messages, $\mathcal{A}$'s view corresponds to Hyb_7.

Now, let's see how $\mathcal{A}_{\pi^{\mathsf{SM}}}$ breaks the security of protocol π^{SM}. For every honest party P_i and malicious party P_j, $\mathcal{A}_{\pi^{\mathsf{SM}}}$ runs the extractor $\mathsf{Ext}_{\mathsf{NMCom}}$ of the non-malleable commitment scheme using the messages in all the "Good" threads that correspond to the non-malleable commitment from P_j to P_i. Let the output of $\mathsf{Ext}_{\mathsf{NMCom}}$ be t_i^*. If for some pair of parties, $\mathsf{TDValid}(\mathsf{td}_1^{i\to j}, \mathsf{t}_i^*) = 1$, then $\mathcal{A}_{\pi^{\mathsf{SM}}}$ outputs case 2 indicating that the messages sent by the challenger were simulated. Else, it outputs case 1 indicating that the messages were generated honestly.

Let's analyze why this works. We know that the invariant doesn't hold in Hyb_7 so there exists honest party P_i^* and malicious party P_j^* such that event E doesn't hold. That is, the adversary P_j^*, using the non-malleable commitment, commits to a valid trapdoor t_i^* for the trapdoor generation messages of the honest party P_i^* with non-negligible probability ϵ. Therefore, since the invariant holds in Hyb_6 with non-negligible probability, when the extractor $\mathsf{Ext}_{\mathsf{NMCom}}$ outputs a valid trapdoor t_i^* corresponding to this pair of parties, it must be the case that we are in Hyb_6 with non-negligible probability. That is, when $\mathsf{Ext}_{\mathsf{NMCom}}$ outputs a valid trapdoor, it corresponds to $\mathcal{A}_{\pi^{\mathsf{SM}}}$ receiving simulated messages and otherwise, it corresponds to $\mathcal{A}_{\pi^{\mathsf{SM}}}$ receiving honestly generated messages. Thus, $\mathcal{A}_{\pi^{\mathsf{SM}}}$ breaks the security of the protocol π^{SM} which is a contradiction. Hence, the invariant holds in Hyb_7 as well. □

Claim 22. *Assuming the security of protocol* π^{SM}*,* Hyb_6 *is indistinguishable from* Hyb_7*.*

Proof. The only difference between Hyb_6 and Hyb_7 is that in Hyb_7, in the main thread, the simulator now computes the messages of protocol π^{SM} using the simulated algorithms $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$. If there exists an adversary $\mathcal{A}$ that can distinguish between the two hybrids, we can use $\mathcal{A}$ to design an adversary $\mathcal{A}_{\pi^{\mathsf{SM}}}$ that breaks the security of the protocol π^{SM}. The rest of the proof is similar to the proof of Claim 21 above.

□

Claim 23. *If the "Check Abort" step succeeds, assuming the bounded rewinding security of the scheme* RWI *and the existence of an extractor* $\mathsf{Ext}_{\mathsf{NMCom}}$ *for the non-malleable commitment scheme* NMCom*, the invariant holds in* Hyb_8*.*

Proof. This is same as the proof of Claim 15. □

Claim 24. *Assuming the bounded rewinding security of the scheme* RWI*,* Hyb_7 *is indistinguishable from* Hyb_8*.*

Proof. This is same as the proof of Claim 16. □

Claim 25. *If the "Check Abort" step succeeds, the invariant holds in* Hyb_9*.*

Proof. There is no difference in the main thread between Hyb_8 and Hyb_9. Also, the new look-ahead threads are identical to Hyb_8 and hence the invariant continues to hold true. □

Claim 26. *Assuming soundness of the argument systems* $\mathsf{WI}, \mathsf{RWI}$ *and* WZK*, the existence of an extractor* Ext *for the extractable commitment scheme* $\mathsf{R.ECom}$ *and the existence of the trapdoor extractor* TDExt *for the trapdoor generation protocol* TDGen*,* Hyb_8 *is indistinguishable from* Hyb_9*.*

Proof. Observe that the only difference between the two hybrids is that the simulator outputs "Special Abort" in the input extraction phase in the second set of look-ahead threads in Hyb_9 and also outputs "Special Abort 2" in Hyb_9.

The proof for the following statement follows exactly as in the proof of *Claim* 10 : $\mathsf{Pr}[\ \mathsf{Sim}_{\mathsf{Hyb}}$ outputs "Special Abort" $] \leq \mathsf{negl}(\lambda)$ in the second set of look-ahead threads.

We now show that the $\mathsf{Pr}[\ \mathsf{Sim}_{\mathsf{Hyb}}$ outputs "Special Abort 2" $] \leq \mathsf{negl}(\lambda)$ in Hyb_9. This happens only if there $\exists\ j$ such that $\{\mathsf{x}_j, \mathsf{r}_j\} \neq \{\mathsf{x}^*_j, \mathsf{r}^*_j\}$. However, we already know that in each set of "Good" look-ahead threads, the algorithm Ext succeeds with noticeable probability. Therefore, from the correctness of Ext, it immediately follows that for all j, $\{\mathsf{x}_j, \mathsf{r}_j\} = \{\mathsf{x}^*_j, \mathsf{r}^*_j\}$ and this completes the proof. □

Claim 27. *Assuming the security of all the primitives used in the construction,*

- *If the "Check Abort" step succeeds, the invariant holds in* Hyb_{10}*.*
- Hyb_9 *is computationally indistinguishable from* Hyb_{10}*.*

Proof. This follows by a repeated application of the proofs of Claim 13, Claim 14, Claim 15, Claim 16, Claim 17, Claim 18, Claim 19, Claim 20, Claim 21, Claim 22, Claim 23, Claim 24. □

Claim 28. Hyb_{10} *is computationally indistinguishable from* Hyb_{11}*.*

Proof. We know that the adversary's input values extracted by using the first set of look-ahead threads is same as those extracted by using the second set of look-ahead threads. Therefore, the only difference between Hyb_{10} and Hyb_{11} is that in Hyb_{10}, after the input extraction, $\mathsf{Sim}_{\mathsf{Hyb}}$ rewinds the main thread $\frac{1}{\mu}$ times while in Hyb_{11}, $\mathsf{Sim}_{\mathsf{Hyb}}$ first estimates the probability of not aborting to be ϵ' and then rewinds the main thread $\mathsf{min}(2^\lambda, \frac{\lambda^2}{\epsilon'})$ times. The rest of the proof follows in a very similar manner to the proof of claim 5.8 in [Lin17]. That is, we show that if the "Check Abort" step succeeds, the simulator fails in Hyb_{11} only with negligible probability using the claim in [Lin17]. We already know that in Hyb_{10}, if the "Check Abort" step succeeds, the simulation successfully forces the output and hence, this completes the proof. □

Observe that Hyb_{11} is the ideal world and this completes the proof of Theorem 7.

7 Simulation Extractable Promise ZK

In this section, we define the notion of Simulation Extractable Promise ZK and then give a 3 round construction of it. Recall the notion of an algorithm $\mathsf{pExtract}$ being "Admissible" from Section 5.

An n-round delayed-input interactive argument $(\mathsf{Prove}, \mathsf{Verify}, \mathsf{Valid})$ in the simultaneous message setting is said to be a simulation-extractable promise ZK argument system if it additionally satisfies the properties of *simulation security* and *simulation-extractability* defined below.

We consider a *synchronous* man-in-the-middle adversary, who completes each round in both the left and right sessions before starting the next round in either session. Note that MIM could be rushing but still has to complete one round in both sessions before proceeding to the next round in either session.

Definition 7 (Simulation-Extractable Promise ZK)**.** *An n-round distributional delayed-input simultaneous-message interactive argument* $(\mathsf{Prove}, \mathsf{Verify}, \mathsf{Valid})$ *where* $\mathsf{Prove} = (P_1, P_2)$ *for a language L is said to be* simulation-extractble promise zero knowledge *if, for every efficiently sampleable distribution $(\mathcal{X}_\lambda, \mathcal{W}_\lambda)$ on R_L, i.e., $\mathsf{Supp}(\mathcal{X}_\lambda, \mathcal{W}_\lambda) = \{(x, w) : x \in L \cap \{0,1\}^\lambda, w \in R_L(x)\}$,*

every non-uniform PPT MIM, every $z \in \{0,1\}^*$, every $c > 0$, and all admissible $\mathsf{pExtract}_c$, there exists an a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that:

Consider the experiments REAL and IDEAL defined below. Let $\mathsf{View}_{\mathsf{MIM}}[\mathsf{REAL}]$ denote the output of the experiment REAL and $\mathsf{View}_{\mathsf{MIM}}[\mathsf{IDEAL}]$ denote the output of the experiment IDEAL. Then:

1. **Simulation Security:** *For every PPT distinguisher $\mathcal{D}$,*
   - *The running time of oracle algorithm $\mathcal{S}_2$ on input (z, x, st, p) is $\mathsf{poly}(\lambda) \cdot O(\frac{1}{p})$.*
   - *Let $\epsilon = \lambda^{-c}$. Then,*

$$\left| \Pr\left[\mathcal{D}(z, \mathsf{View}_{\mathsf{MIM}}[\mathsf{REAL}] = 1\right] - \Pr\left[\mathcal{D}(z, \mathsf{View}_{\mathsf{MIM}}[\mathsf{IDEAL}]) = 1\right] \right| \leq \epsilon$$

   where the probability is over the random coins of the parties in the below experiments.

2. **Simulation Extractability:** *For every $x^* \in \{0,1\}^\lambda$ chosen by MIM in the right session adaptively depending upon the first $(n-2)$ rounds:*

$$\Pr\left[E^{\mathsf{MIM}}(z, \mathsf{Trans}) = w \wedge R(x^*, w) = 1 \;\middle|\; \langle \mathcal{S}(x, z), \mathsf{MIM}(z)\rangle_R = 1\right] \geq (1 - \mathsf{negl}(\lambda)),$$

   where the probability is over the random coins of all the parties. $\langle \mathsf{Sim}(x, z), \mathsf{MIM}(z)\rangle_R$ denotes the output of Sim in the right session.

Experiment REAL**:** *. A man-in-the-middle adversary* MIM *interacts with an honest prover* P *in the left session and an honest verifier* V *in the right session. Prover* P *has input* $(\mathcal{X}_\lambda, \mathcal{W}_\lambda)$ *- an efficiently sampleable distribution on* R_L*. Let* z *be the auxiliary input given to* MIM *and let* (x, w) *be the instance-witness pair sampled by* P *in the interaction. The experiment proceeds as follows:*

1. *In the left session:*
   - *Compute* $(\mathsf{msg}, \mathsf{st}) \leftarrow P_1(1^\lambda)$*. Output* msg*.*
   - *Then, sample* $(x, w) \leftarrow (\mathcal{X}_\lambda, \mathcal{W}_\lambda)$*.*
   - *Output* $\langle P_2(x, w, \mathsf{st}), \mathsf{MIM}\rangle$*.*
2. *In the right session, output* $\langle \mathsf{MIM}, V(1^\lambda)\rangle$*.*

Experiment IDEAL**:** *. A man-in-the-middle adversary* MIM *interacts with a simulator* Sim *in both the left and right sessions. The simulator* $\mathsf{Sim} = (P_1, \mathcal{S}_2)$ *has input* $(\mathcal{X}_\lambda, \mathcal{W}_\lambda)$ *- an efficiently sampleable distribution on* R_L*. Additionally,* Sim *is given as input a statement* x *where* (x, w) *is sampled from the distribution* $(\mathcal{X}_\lambda, \mathcal{W}_\lambda)$ *but the corresponding witness* w *is not given to* Sim*. In this interaction, let* z *be the auxiliary input given to* MIM*.*

- *Compute* $(\mathsf{msg}, \mathsf{st}) \leftarrow P_1(1^\lambda)$*. Output* msg*.*
- *Let* $\mathsf{pExtract}_c^{\mathsf{MIM}}(\mathsf{msg}, \mathsf{st}) = p$*.*
- *If* $p = 0$*, sample* $(x, w) \leftarrow (\mathcal{X}_\lambda, \mathcal{W}_\lambda)$ *and output* $\langle P_2(x, w, \mathsf{st}), \mathsf{MIM}\rangle$ *in the left session and* $\langle \mathsf{MIM}, V(1^\lambda)\rangle$ *in the right session.*
- *Else, sample* $x \leftarrow (\mathcal{X}_\lambda)$ *and output* $\mathcal{S}_2^{\mathsf{MIM}}(z, x, \mathsf{st}, p)$*.*

Remark: Note that the simulation security property implies the distributional promise ZK property defined in the previous section. Similarly, simulation-extractability property implies soundness.

7.1 Construction

In this section, we construct a three round simulation-extractable promise ZK argument system. Formally, we prove the following theorem:

Theorem 9. *Assuming the existence of polynomially secure injective one way functions, the protocol* $\pi^{\mathsf{SE-PZK}}$ *is a three round simulation-extractable promise ZK argument.*

We start by describing some notation and cryptographic primitives that we use in our construction

Building Blocks. Our construction relies on the following list of cryptographic primitives.

- $\mathsf{TDGen} = (\mathsf{TDGen}_1, \mathsf{TDGen}_2, \mathsf{TDGen}_3, \mathsf{TDOut})$ is the three-message trapdoor generation protocol as defined in Section 4. The first 3 algorithms are used to generate the messages of the protocol while TDOut checks that the execution was honest. TDGen also has two associated PPT algorithms $\mathsf{TDValid}, \mathsf{TDExt}$.
  Recall that the algorithm $\mathsf{TDValid}$ takes as input a tuple of 4 values : the first three being outputs of the algorithms $\mathsf{TDGen}_1, \mathsf{TDGen}_2, \mathsf{TDGen}_3$ and outputs 1 if the fourth value is a

valid trapdoor with respect to these three.
The algorithm TDExt, as defined earlier, with overwhelming probability, outputs a valid trapdoor if given honestly generated messages of the trapdoor generator in 3 executions with a common first round.

- $\mathsf{RWI} = (\mathsf{RWI}_1, \mathsf{RWI}_2, \mathsf{RWI}_3, \mathsf{RWI}_4)$ is the three round delayed-input witness indistinguishable argument with bounded rewinding security defined in Section 4. The fourth algorithm RWI_4 is the final verification algorithm. We will set the rewinding security parameter L to be 5 in all our applications.

- $\mathsf{NMCom} = (\mathsf{NMCom}_1, \mathsf{NMCom}_2, \mathsf{NMCom}_3)$ is any three-message delayed-input non-malleable commitment scheme with respect to commitment in which the third round message is indistinguishable from a random string when the input is $\perp$. Let $\mathsf{Ext}_{\mathsf{NMCom}}$ denote the polynomial time extractor of this non-malleable commitment scheme. We require that given the transcript of 2 executions of the protocol, $\mathsf{Ext}_{\mathsf{NMCom}}$ can extract the value committed to inside the commitment with non-negligible probability. Also, given the transcript of polynomial many executions of the protocol, $\mathsf{Ext}_{\mathsf{NMCom}}$ can extract the value committed to inside the commitment except with negligible probability.

  The non-malleable commitment scheme defined in Goyal et al.[GPR16] is one such example that can be based on injective one way functions.

  Remark: The NMCom we use is tagged. In the authenticated channels setting, the tag of each user performing a non-malleable commitment can just be its identity. In the general setting, in the first round, each party can choose a strong digital signature verification key VK and signing key, and then sign all its messages using this signature scheme for every message sent in the protocol. This VK is then used as the tag. This ensures that every adversarial party must choose a tag that is different from any tags chosen by honest parties, otherwise the adversary will not be able to sign any of its messages by the existential unforgeability property of the signature scheme. This is precisely the property that is assumed when applying NMCom. For ease of notation, we suppress writing the tags explicitly in our protocols below.

NP Languages. In our construction, to decide any NP language L characterized by relation R, we define a new NP language L_{RWI} characterized by the following relation R_{RWI}.
Statement: $\mathsf{st}_{\mathsf{RWI}} = (x, \mathsf{c}_{a,1}, \mathsf{c}_{a,2}, \mathsf{c}_{a,3}, \mathsf{c}_{b,1}, \mathsf{c}_{b,2}, \mathsf{c}_{b,3}, \mathsf{c}_{t,1}, \mathsf{c}_{t,2}, \mathsf{c}_{t,3}, \mathsf{td}_1, \mathsf{td}_2, \mathsf{td}_3)$
Witness: $\mathsf{w}_{\mathsf{RWI}} = (w, \mathsf{r}_a, \mathsf{r}_b, \mathsf{t}, \mathsf{r}_t)$
Relation: $R(\mathsf{st}_{\mathsf{RWI}}, \mathsf{w}_{\mathsf{RWI}}) = 1$ if and only if :

- $\mathsf{c}_{a,1} = \mathsf{NMCom}_1(\mathsf{r}_a)$ AND
- $\mathsf{c}_{a,3} = \mathsf{NMCom}_3(w, \mathsf{c}_{a,1}, \mathsf{c}_{a,2}; \mathsf{r}_a)$ AND
- $R(x, w) = 1$

  (OR)

- $\mathsf{c}_{b,1} = \mathsf{NMCom}_1(\mathsf{r}_b)$ AND
- $\mathsf{c}_{b,3} = \mathsf{NMCom}_3(w, \mathsf{c}_{b,1}, \mathsf{c}_{b,2}; \mathsf{r}_b)$ AND
- $R(x, w) = 1$

  (OR)

- $\mathsf{TDValid}(\mathsf{td}_1, \mathsf{td}_2, \mathsf{td}_3, \mathsf{t}) = 1$ AND
- $\mathsf{c}_{t,1} = \mathsf{BRew.ECom}_1(\mathsf{r}_t)$ AND
- $\mathsf{c}_{t,3} = \mathsf{BRew.ECom}_3(\mathsf{t}, \mathsf{c}_{t,1}, \mathsf{c}_{t,2}; \mathsf{r}_t)$.

That is, either :

1. x is in the language L with witness w that is committed to using the first non-malleable commitment $(\mathsf{c}_{a,1}, \mathsf{c}_{a,2}, \mathsf{c}_{a,3})$ (OR)
2. x is in the language L with witness w that is committed to using the second non-malleable commitment $(\mathsf{c}_{b,1}, \mathsf{c}_{b,2}, \mathsf{c}_{b,3})$ (OR)
3. the third non-malleable commitment $(\mathsf{c}_{t,1}, \mathsf{c}_{t,2}, \mathsf{c}_{t,3})$ is to a value t that is a valid trapdoor for the messages $(\mathsf{td}_1, \mathsf{td}_2, \mathsf{td}_3)$ generated using the trapdoor generation algorithms.

7.1.1 The Protocol

Let P and V denote the prover and verifier, respectively. Let L be any NP language with an associated relation R_L. Let $(\mathcal{X}_\lambda, \mathcal{W}_\lambda)$ be any efficiently sampleable distribution on R_L.

We construct a three round protocol $\pi^{\mathsf{SE-PZK}} = (P, V, \mathsf{Valid})$ for L. The protocol is described in Figure Figure 7. We first describe the algorithms (P, V) denoting the interaction between the prover and verifier. The description of algorithm Valid is given at the end. We use the notation $P \to V$ in the superscript to denote that the message was sent by P to V. The round number of any sub-protocol being used is written in the subscript. We use three instantiations of the non-malleable commitment scheme NMCom; we use $(a, i), (b, i), (t, i)$ in the subscript to denote each of the instantiations (where i is the round number).

Completeness follows from the correctness of the underlying primitives used. We will now describe the security proof.

7.2 Security Proof

We first give a description of the simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ only in the interaction with the adversary MIM in the left session. We then augment Sim with the honest verifier's strategy for the right session to construct the final simulator $\mathsf{Sim}_{\mathsf{SE}}$. That is, our ideal world simulator $\mathsf{Sim}_{\mathsf{SE}}$, in the left session, performs exactly as done by Sim and in the right session, performs the role of an honest verifier.

Before we describe the details, lets recall the basic strategy followed by a rewinding simulator. Sim creates a "main thread" of execution that will be actually output at the end of the simulation and three "look-ahead" threads that will facilitate the extraction of the adversary's trapdoor. Each look-ahead thread created by Sim shares the first round with the main thread, but contains different messages in the second and third rounds. Sim will use the adversary's messages in the third round of these look-ahead threads to extract the adversary's trapdoor and then use this appropriately in the main thread to simulate the adversary's final view. Note that in the final simulation, for each look-ahead thread, $\mathsf{Sim}_{\mathsf{SE}}$ plays the role of the honest verifier in the right session.

Running time of Sim_2: The number of look-ahead threads created is $(\lambda \cdot \frac{1}{p})$ where p is the value that is given as input to Sim_2. Hence, it is easy to see that the running time of Sim_2 is indeed $\mathsf{poly}(\lambda) \cdot O(\frac{1}{p})$.

Inputs: Prover P has input $(\mathcal{X}_\lambda, \mathcal{W}_\lambda)$ - an efficiently sampleable distribution on R_L.

1. **Round 1:**
   - **Prover message:**
     Compute and send $(\mathsf{rwi}_1^{P\to V}) \leftarrow \mathsf{RWI}_1(1^\lambda)$, $\mathsf{c}_{a,1}^{P\to V} \leftarrow \mathsf{NMCom}_1(\mathsf{r}_a)$, $\mathsf{c}_{b,1}^{P\to V} \leftarrow \mathsf{NMCom}_1(\mathsf{r}_b)$ and $\mathsf{c}_{t,1}^{P\to V} \leftarrow \mathsf{NMCom}_1(\mathsf{r}_t)$ using random strings $\mathsf{r}_a, \mathsf{r}_b, \mathsf{r}_t$.
   - **Verifier message:**
     Compute and send $\mathsf{td}_1^{V\to P} \leftarrow \mathsf{TDGen}_1(\mathsf{r}_{\mathsf{td}})$ using a random string r_{td}.
2. **Round 2:**
   - **Prover message:**
     - Sample $(x = (x_1, x_2), w) \leftarrow (\mathcal{X}_\lambda, \mathcal{W}_\lambda)$.
     - Compute and send $\mathsf{td}_2^{P\to V} \leftarrow \mathsf{TDGen}_2(\mathsf{td}_1^{P\to V})$ along with x_1.
   - **Verifier message:**
     Compute and send $\mathsf{rwi}_2^{V\to P} \leftarrow \mathsf{RWI}_2(\mathsf{wi}_1^{P\to V})$, $\mathsf{c}_{a,2}^{V\to P} \leftarrow \mathsf{NMCom}_2(\mathsf{c}_{a,1}^{P\to V})$, $\mathsf{c}_{b,2}^{V\to P} \leftarrow \mathsf{NMCom}_2(\mathsf{c}_{b,1}^{P\to V})$, $\mathsf{c}_{t,2}^{V\to P} \leftarrow \mathsf{NMCom}_2(\mathsf{c}_{t,1}^{P\to V})$.
3. **Round 3:**
   - **Prover message:**
     - Compute $\mathsf{c}_{a,3}^{P\to V} \leftarrow \mathsf{NMCom}_3(w, \mathsf{c}_{a,1}^{P\to V}, \mathsf{c}_{a,2}^{V\to P}; \mathsf{r}_a)$, $\mathsf{c}_{b,3}^{P\to V} \leftarrow \mathsf{NMCom}_3(\bot, \mathsf{c}_{b,1}^{P\to V}, \mathsf{c}_{b,2}^{V\to P}; \mathsf{r}_b)$ and $\mathsf{c}_{t,3}^{P\to V} \leftarrow \mathsf{NMCom}_3(\bot, \mathsf{c}_{t,1}^{P\to V}, \mathsf{c}_{t,2}^{V\to P}; \mathsf{r}_t)$.
     - Generate $\mathsf{rwi}_3^{P\to V} \leftarrow \mathsf{RWI}_3(\mathsf{rwi}_1^{P\to V}, \mathsf{rwi}_2^{V\to P}, \mathsf{st}_{\mathsf{RWI}}, \mathsf{w}_{\mathsf{RWI}})$ for the statement $\mathsf{st}_{\mathsf{RWI}} = (x, \mathsf{c}_{a,1}^{P\to V}, \mathsf{c}_{a,2}^{V\to P}, \mathsf{c}_{a,3}^{P\to V}, \mathsf{c}_{b,1}^{P\to V}, \mathsf{c}_{b,2}^{V\to P}, \mathsf{c}_{b,3}^{P\to V}, \mathsf{c}_{t,1}^{P\to V}, \mathsf{c}_{t,2}^{V\to P}, \mathsf{c}_{t,3}^{P\to V}, \mathsf{td}_1^{V\to P}, \mathsf{td}_2^{P\to V}, \mathsf{td}_3^{V\to P})$ using witness $(w, \mathsf{r}_a, \bot, \bot, \bot)$.
     - Send $(x_2, \mathsf{c}_{a,3}^{P\to V}, \mathsf{c}_{b,3}^{P\to V}, \mathsf{c}_{t,3}^{P\to V}, \mathsf{rwi}_3^{P\to V})$.
   - **Verifier message:**
     Compute and send $\mathsf{td}_3^{V\to P} \leftarrow \mathsf{TDGen}_3(\mathsf{td}_1^{V\to P}, \mathsf{td}_2^{P\to V}; \mathsf{r}_{\mathsf{td}})$ using randomness r_{td}.
4. **Verifier Output:**
   Output $\mathsf{RWI}_4(\mathsf{rwi}_1^{P\to V}, \mathsf{rwi}_2^{V\to P}, \mathsf{rwi}_3^{P\to V}, \mathsf{st}_{\mathsf{RWI}})$.

Valid(Trans):
Given the transcript of the protocol execution, output 1 if $\mathsf{TDOut}(\mathsf{td}_1^{V\to P}, \mathsf{td}_2^{P\to V}, \mathsf{td}_3^{V\to P}) = 1$.

Figure 7: Three round Simulation-Extractable Promise ZK argument.

To prevent a cluttered description, we overload notation when referring to the same object in the main thread and the look-ahead threads. However, it will be clear from context which thread's object is being referred to.

The description of Sim is given in Figure 8.

7.2.1 Hybrids

We will now prove both the security properties by using a sequence of hybrids. The first hybrid Hyb_0 will correspond to the real world experiment where the adversary MIM interacts with an honest prover P on the left and honest verifier V on the right. The last hybrid Hyb_5 corresponds to the ideal world where MIM interacts with the simulator $\mathsf{Sim}_{\mathsf{SE}}$. Additionally, we will also maintain

$\mathsf{Sim}_1(z)$**:** Recall that by definition $\mathsf{Sim}_1(z) = P_1(1^\lambda)$. That is, it does the following:

- Compute and send $(\mathsf{rwi}_1^{P\to V}) \leftarrow \mathsf{RWI}_1(1^\lambda)$, $\mathsf{c}_{a,1}^{P\to V} \leftarrow \mathsf{NMCom}_1(\mathsf{r}_a)$, $\mathsf{c}_{b,1}^{P\to V} \leftarrow \mathsf{NMCom}_1(\mathsf{r}_b)$ and $\mathsf{c}_{t,1}^{P\to V} \leftarrow \mathsf{NMCom}_1(\mathsf{r}_t)$ using random strings $\mathsf{r}_a, \mathsf{r}_b, \mathsf{r}_t$. Note that this denotes the output msg with the associated state st being $(\mathsf{r}_a, \mathsf{r}_b, \mathsf{r}_t)$ and the randomness used to generate $\mathsf{rwi}_1^{P\to V}$.
- Receive $\mathsf{td}_1^{V\to P}$ from V^*.

$\mathsf{Sim}_2(z, x, \mathsf{st}, p)$:
Let $p = \mathsf{pExtract}_c^{\mathsf{MIM}}(\mathsf{msg}, \mathsf{st})$. If $p = 0$, sample $(x, w) \leftarrow (\mathcal{X}_\lambda, \mathcal{W}_\lambda)$ and output $\langle P_2(x, w, \mathsf{st}), \mathsf{MIM}\rangle$. Else:

1. **Round 2:**
   - Create a set of $(\lambda \cdot \frac{1}{p})$ look-ahead threads that run only rounds 2 and 3 of the protocol.
   - In each look-ahead thread, sample $(x = (x_1, x_2), w) \leftarrow (\mathcal{X}_\lambda, \mathcal{W}_\lambda)$.
   - Then, in each of the threads (main and look-ahead) :
     - Compute and send $\mathsf{td}_2^{P\to V} \leftarrow \mathsf{TDGen}_2(\mathsf{td}_1^{P\to V})$ along with x_1.
     - Receive $\mathsf{rwi}_2^{V\to P}, \mathsf{c}_{a,2}^{V\to P}, \mathsf{c}_{b,2}^{V\to P}$ and $\mathsf{c}_{t,2}^{V\to P}$.
2. **Round 3:**
   In each look-ahead thread:
   - Compute $\mathsf{c}_{a,3}^{P\to V} \leftarrow \mathsf{NMCom}_3(\bot, \mathsf{c}_{a,1}^{P\to V}, \mathsf{c}_{a,2}^{V\to P}; \mathsf{r}_a)$, $\mathsf{c}_{b,3}^{P\to V} \leftarrow \mathsf{NMCom}_3(w, \mathsf{c}_{b,1}^{P\to V}, \mathsf{c}_{b,2}^{V\to P}; \mathsf{r}_b)$ and $\mathsf{c}_{t,3}^{P\to V} \leftarrow \mathsf{NMCom}_3(\bot, \mathsf{c}_{t,1}^{P\to V}, \mathsf{c}_{t,2}^{V\to P}; \mathsf{r}_t)$.
   - Generate $\mathsf{rwi}_3^{P\to V} \leftarrow \mathsf{RWI}_3(\mathsf{rwi}_1^{P\to V}, \mathsf{rwi}_2^{V\to P}, \mathsf{st}_{\mathsf{RWI}}, \mathsf{w}_{\mathsf{RWI}})$ for the statement $\mathsf{st}_{\mathsf{RWI}} = (x, \mathsf{c}_{a,1}^{P\to V}, \mathsf{c}_{a,2}^{V\to P}, \mathsf{c}_{a,3}^{P\to V}, \mathsf{c}_{b,1}^{P\to V}, \mathsf{c}_{b,2}^{V\to P}, \mathsf{c}_{b,3}^{P\to V}, \mathsf{c}_{t,1}^{P\to V}, \mathsf{c}_{t,2}^{V\to P}, \mathsf{c}_{t,3}^{P\to V}, \mathsf{td}_1^{V\to P}, \mathsf{td}_2^{P\to V}, \mathsf{td}_3^{V\to P})$ using witness $(w, \bot, \mathsf{r}_b, \bot, \bot)$.
   - Send $(x_2, \mathsf{rwi}_3^{P\to V})$ and receive $\mathsf{td}_3^{V\to P}$.

   Input Extraction:
   - Run the trapdoor extractor using the trapdoor generation messages of all the look-ahead threads. That is, compute $\mathsf{t}_V \leftarrow \mathsf{TDExt}(\mathsf{td}_1^{V\to P}, \{\mathsf{td}_2^{P\to V}, \mathsf{td}_3^{V\to P}\})$ where the set denotes the pair of values from all the look-ahead threads.
   - Output "Special Abort" if TDExt fails.

   Main thread:
   - Compute $\mathsf{c}_{a,3}^{P\to V} \leftarrow \mathsf{NMCom}_3(\bot, \mathsf{c}_{a,1}^{P\to V}, \mathsf{c}_{a,2}^{V\to P}; \mathsf{r}_a)$, $\mathsf{c}_{b,3}^{P\to V} \leftarrow \mathsf{NMCom}_3(\bot, \mathsf{c}_{b,1}^{P\to V}, \mathsf{c}_{b,2}^{V\to P}; \mathsf{r}_b)$ and $\mathsf{c}_{t,3}^{P\to V} \leftarrow \mathsf{NMCom}_3(\mathsf{t}_V, \mathsf{c}_{t,1}^{P\to V}, \mathsf{c}_{t,2}^{V\to P}; \mathsf{r}_t)$.
   - Generate $\mathsf{rwi}_3^{P\to V} \leftarrow \mathsf{RWI}_3(\mathsf{rwi}_1^{P\to V}, \mathsf{rwi}_2^{V\to P}, \mathsf{st}_{\mathsf{RWI}}, \mathsf{w}_{\mathsf{RWI}})$ for the statement $\mathsf{st}_{\mathsf{RWI}} = (x, \mathsf{c}_{a,1}^{P\to V}, \mathsf{c}_{a,2}^{V\to P}, \mathsf{c}_{a,3}^{P\to V}, \mathsf{c}_{b,1}^{P\to V}, \mathsf{c}_{b,2}^{V\to P}, \mathsf{c}_{b,3}^{P\to V}, \mathsf{c}_{t,1}^{P\to V}, \mathsf{c}_{t,2}^{V\to P}, \mathsf{c}_{t,3}^{P\to V}, \mathsf{td}_1^{V\to P}, \mathsf{td}_2^{P\to V}, \mathsf{td}_3^{V\to P})$ using witness $(\bot, \bot, \bot, \mathsf{t}_V, \mathsf{r}_t)$.
   - Send $(x_2, \mathsf{rwi}_3^{P\to V})$ and receive $\mathsf{td}_3^{V\to P}$.

Figure 8: Simulator's description.

a couple of invariants that will be useful in the proof.

- Hyb_0 **- Real World:** In this hybrid, consider a simulator $\mathsf{Sim}_{\mathsf{Hyb}}$ that plays the role of the

honest prover on the left session and an honest verifier on the right session.

- Hyb_1 **- Extraction:** In this hybrid, $\mathsf{Sim}_{\mathsf{Hyb}}$ first runs the protocol honestly. If the adversary aborts at the end of the execution, then $\mathsf{Sim}_{\mathsf{Hyb}}$ stops here. Else, $\mathsf{Sim}_{\mathsf{Hyb}}$ rewinds back to the end of round 1 on the left session and creates a fresh set of $(\lambda \cdot \frac{1}{\epsilon})$ look-ahead threads on the left session. In all the look-ahead threads, $\mathsf{Sim}_{\mathsf{Hyb}}$ performs exactly as described in Hyb_1. Additionally, using the messages in the left session, $\mathsf{Sim}_{\mathsf{Hyb}}$ also runs the "Input Extraction" phase described in round 3 of the description of Sim to extract the trapdoor t_V. Finally, $\mathsf{Sim}_{\mathsf{Hyb}}$ completes the main thread on the left session by running the honest prover strategy exactly as in Hyb_0.

- Hyb_2 **- Changing Commitment to Trapdoor:** In the main thread, on the left session, $\mathsf{Sim}_{\mathsf{Hyb}}$ does the following: in round 3, compute $\mathsf{c}_{t,3}^{P\to V} \leftarrow \mathsf{NMCom}_3(\mathsf{t}_V, \mathsf{c}_{t,1}^{P\to V}, \mathsf{c}_{t,2}^{V\to P}; \mathsf{r}_{\mathsf{c}})$.

- Hyb_3 **- Switching bounded rewinding secure WI proofs:** In the main thread, on the left session, $\mathsf{Sim}_{\mathsf{Hyb}}$ does the following: $\mathsf{rwi}_3^{P\to V} \leftarrow \mathsf{RWI}_3(\mathsf{rwi}_1^{P\to V}, \mathsf{rwi}_2^{V\to P}, \mathsf{st}_{\mathsf{RWI}}, \mathsf{w}_{\mathsf{RWI}})$ for the statement $\mathsf{st}_{\mathsf{RWI}} = (x, \mathsf{c}_{a,1}^{P\to V}, \mathsf{c}_{a,2}^{V\to P}, \mathsf{c}_{a,3}^{P\to V}, \mathsf{c}_{b,1}^{P\to V}, \mathsf{c}_{b,2}^{V\to P}, \mathsf{c}_{b,3}^{P\to V}, \mathsf{c}_{t,1}^{P\to V}, \mathsf{c}_{t,2}^{V\to P}, \mathsf{c}_{t,3}^{P\to V}, \mathsf{td}_1^{V\to P}, \mathsf{td}_2^{P\to V}, \mathsf{td}_3^{V\to P})$ using witness $(\bot, \bot, \bot, \mathsf{t}_V, \mathsf{r}_t)$.

- Hyb_4 **- Changing Commitment to Witness:** In the main thread, on the left session, $\mathsf{Sim}_{\mathsf{Hyb}}$ does the following: in round 3, compute $\mathsf{c}_{a,3}^{P\to V} \leftarrow \mathsf{NMCom}_3(\bot, \mathsf{c}_{a,1}^{P\to V}, \mathsf{c}_{a,2}^{V\to P}; \mathsf{r}_a)$.

- Hyb_5 **- Using** $\mathsf{pExtract}_c$**:** In this hybrid, the number of look-ahead threads created is $(\lambda \cdot \frac{1}{p})$. Also, on the left session, $\mathsf{Sim}_{\mathsf{Hyb}}$ no longer samples (x, w) in round 3 and instead uses the external input x. The description of $\mathsf{Sim}_{\mathsf{Hyb}}$ in this hybrid matches the description of the simulator $\mathsf{Sim}_{\mathsf{SE}}$.

7.2.2 Invariants

We now describe the two invariants.

Definition 8 (Invariant T)**.** *Consider the right session between* MIM *and* $\mathsf{Sim}_{\mathsf{Hyb}}$*.* $\mathsf{td}_1^{V\to P}$ *denotes the first message of the trapdoor generation protocol with* $\mathsf{Sim}_{\mathsf{Hyb}}$ *as the trapdoor generator. The values* $(\mathsf{c}_{t,1}^{P\to V}, \mathsf{c}_{t,2}^{V\to P}, \mathsf{c}_{t,3}^{P\to V})$ *denote the messages of the non-malleable commitment with* MIM *as the creator.*
Consider the following event E_t *which occurs if* $\exists(\mathsf{t}_i, \mathsf{r}_t)$ *such that:*

- $\mathsf{c}_{t,1}^{P\to V} = \mathsf{NMCom}_1(\mathsf{r}_t)$ *(AND)*
- $\mathsf{c}_{t,3}^{P\to V} = \mathsf{NMCom}_3(\mathsf{t}_i, \mathsf{c}_{t,1}^{P\to V}, \mathsf{c}_{t,2}^{V\to P}; \mathsf{r}_t)$*. (AND)*
- $\mathsf{TDValid}(\mathsf{td}_1^{V\to P}, \mathsf{t}_i) = 1$*.*

That is, the event E_t *occurs if, in the right session, the adversary* MIM*, using the non-malleable commitment, commits to a valid trapdoor* t_i *for the trapdoor generation messages of* $\mathsf{Sim}_{\mathsf{Hyb}}$*.*

The invariant is : $\mathsf{Pr}[$ *Event* E_t *occurs* $] \leq \mathsf{negl}(\lambda)$*.*

Definition 9 (Invariant W)**.** *Consider the right session between* MIM *and* $\mathsf{Sim}_{\mathsf{Hyb}}$*. The values* $(\mathsf{c}_{a,1}^{P\to V}, \mathsf{c}_{a,2}^{V\to P}, \mathsf{c}_{a,3}^{P\to V})$ *and* $(\mathsf{c}_{b,1}^{P\to V}, \mathsf{c}_{b,2}^{V\to P}, \mathsf{c}_{b,3}^{P\to V})$ *denote the messages of the two non-malleable commitments with* MIM *as the creator. Let* x *be the statement output by* MIM *in the last round.*
Consider the following event E_w *which occurs if* $\exists(w, \mathsf{r}_w)$ *such that:*

- $\mathsf{c}_{a,1}^{P\to V} = \mathsf{NMCom}_1(\mathsf{r}_w)$ *(AND)*
- $\mathsf{c}_{a,3}^{P\to V} = \mathsf{NMCom}_3(w, \mathsf{c}_{a,1}^{P\to V}, \mathsf{c}_{a,2}^{V\to P}; \mathsf{r}_w)$. *(AND)*
- $R(x, w) = 1$.

(OR)

- $\mathsf{c}_{b,1}^{P\to V} = \mathsf{NMCom}_1(\mathsf{r}_w)$ *(AND)*
- $\mathsf{c}_{b,3}^{P\to V} = \mathsf{NMCom}_3(w, \mathsf{c}_{b,1}^{P\to V}, \mathsf{c}_{b,2}^{V\to P}; \mathsf{r}_w)$. *(AND)*
- $R(x, w) = 1$.

That is, the event E_w *occurs if, in the right session, the adversary* MIM*, using either of the first two non-malleable commitments, commits to a witness* w *for the statement* x*.*

The invariant is : $\mathsf{Pr}[\langle \mathsf{Sim}(x, z), \mathsf{MIM}(z)\rangle_R = 1 \wedge$ *Event* E_w *does not occur* $] \leq \mathsf{negl}(\lambda)$.

That is, if the MIM *produces an accepting proof in the right session, then Event* E_w *must occur except with negligible probability.*

7.2.3 Simulation Security

In this section, we will show that Hyb_0 is computationally indistinguishable from Hyb_5 thus proving the Simulation Security property. Along the way, we will also prove that the two invariants hold in each hybrid. This will help us in proving the Simulation Extractability property in the next hybrid.

First, assume by contradiction that there exists an adversary $\mathcal{A}$ that can distinguish the real and ideal worlds with some probability greater than ϵ where $\epsilon > \lambda^{-c}$ for some constant $c > 0$. Then it must be the case that in the ideal world, $p = \mathsf{pExtract}_c^{\mathsf{MIM}}(\mathsf{msg}, \mathsf{st}) \neq 0$. This is because, when $p = 0$, the real and ideal views are identical. Therefore, in the other case, we have $q_{\mathsf{msg}} > p > \epsilon$.

We now prove that every pair of consecutive hybrids is indistinguishable except with probability at most $\frac{\epsilon}{10}$ and this completes the proof.

Claim 29. *Assuming the "1-rewinding security" of the trapdoor generation protocol* TDGen *and the existence of an extractor* $\mathsf{Ext}_{\mathsf{NMCom}}$ *for the non-malleable commitment scheme* NMCom*, Invariant* T *holds in* Hyb_0*.*

Proof. We will prove this by contradiction. Assume that the invariant doesn't hold in Hyb_0. That is, there exists an adversary MIM such that it causes event E_t to occur with non-negligible probability p. We will use this adversary to design an adversary $\mathcal{A}_{\mathsf{TDGen}}$ that breaks the "1-rewinding security" of the trapdoor generation protocol TDGen as defined in Section 4 with non-negligible probability.

$\mathcal{A}_{\mathsf{TDGen}}$ interacts with a challenger $\mathcal{C}_{\mathsf{TDGen}}$ and receives a first round message td_1 corresponding to the protocol TDGen. $\mathcal{A}_{\mathsf{TDGen}}$ performs the role of $\mathsf{Sim}_{\mathsf{Hyb}}$ in its interaction with MIM exactly as done in Hyb_0. Now, in round 1 of the right session, $\mathcal{A}_{\mathsf{TDGen}}$ sets $\mathsf{td}_1^{V\to P}$ as td_1 received from $\mathcal{C}_{\mathsf{TDGen}}$. On receiving a value $\mathsf{td}_2^{P\to V}$ from MIM in round 2 of the right session, $\mathcal{A}_{\mathsf{TDGen}}$ forwards this message to $\mathcal{C}_{\mathsf{TDGen}}$ as its second round message for the protocol TDGen. $\mathcal{A}_{\mathsf{TDGen}}$ receives td_3 from $\mathcal{C}_{\mathsf{TDGen}}$ which is set as $\mathsf{td}_3^{V\to P}$ in its interaction with MIM. $\mathcal{A}_{\mathsf{TDGen}}$ continues with the rest of protocol exactly as in Hyb_0. Then, $\mathcal{A}_{\mathsf{TDGen}}$ rewinds the adversary MIM back to the beginning of round 2. To be consistent with our earlier terminology, this can be interpreted as follows: in the security proof, $\mathcal{A}_{\mathsf{TDGen}}$ creates a look-ahead thread that runs only rounds 2 and 3 of protocol

$\pi^{\mathsf{SE-PZK}}$. Note that this look-ahead thread exists only in the proof of the invariant and not in the description of Hyb_0. As in the main thread, $\mathcal{A}_{\mathsf{TDGen}}$ forwards the adversary's message $\mathsf{td}_2^{P\to V}$ from MIM in round 2 to $\mathcal{C}_{\mathsf{TDGen}}$ and receives td_3 from $\mathcal{C}_{\mathsf{TDGen}}$ which is set as $\mathsf{td}_3^{V\to P}$ in its interaction with MIM. $\mathcal{A}_{\mathsf{TDGen}}$ continues with the rest of protocol $\pi^{\mathsf{SE-PZK}}$ exactly as in Hyb_0.

Now, $\mathcal{A}_{\mathsf{TDGen}}$ runs the extractor $\mathsf{Ext}_{\mathsf{NMCom}}$ of the non-malleable commitment scheme using the messages in both the threads that correspond to the non-malleable commitment from MIM in the right session. Let the output of $\mathsf{Ext}_{\mathsf{NMCom}}$ be t^*. $\mathcal{A}_{\mathsf{TDGen}}$ outputs t^* as a valid trapdoor to $\mathcal{C}_{\mathsf{TDGen}}$.

Let's analyze why this works. We know that event E_t doesn't hold. That is, the adversary MIM, using the non-malleable commitment, commits to a valid trapdoor t_i^* for the trapdoor generation messages of $\mathsf{Sim}_{\mathsf{Hyb}}$ in the right session with non-negligible probability p. Let's say the extractor $\mathsf{Ext}_{\mathsf{NMCom}}$ is successful with non-negligible probability p'. Therefore, with non-negligible probability $p \cdot p'$, $\mathcal{A}_{\mathsf{TDGen}}$ outputs t^* as a valid trapdoor to $\mathcal{C}_{\mathsf{TDGen}}$ which breaks the security of the trapdoor generation protocol TDGen. Thus, it must be the case that the invariant holds in Hyb_0. □

Claim 30. *Assuming the soundness of the bounded rewinding secure WI argument system, Invariant W holds in* Hyb_0.

Proof. We already know that Invariant T holds in Hyb_0. Suppose Invariant W doesn't hold. That is, there exists an adversary MIM such that, with non-negligible probability p in the right session, it produces an accepting SWI argument without committing to a valid witness for the statement in either of the first two non-malleable commitments. We will use MIM to design an adversary $\mathcal{A}_{\mathsf{RWI}}$ that breaks the soundness of the SWI argument system with non-negligible probability.

$\mathcal{A}_{\mathsf{RWI}}$ performs the role of $\mathsf{Sim}_{\mathsf{Hyb}}$ in its interaction with MIM exactly as done in Hyb_0. $\mathcal{A}_{\mathsf{RWI}}$ also interacts with a challenger $\mathcal{C}_{\mathsf{RWI}}$ (who is an honest verifier) in the soundness experiment for the SWI argument system. $\mathcal{A}_{\mathsf{RWI}}$ forwards every message from MIM in the right session to $\mathcal{C}_{\mathsf{RWI}}$ and vice versa.

Now, $\mathcal{A}_{\mathsf{RWI}}$ has broken the soundness of the SWI system with non-negligible probability p. Let's analyze why. In the right session, since invariant T holds, there doesn't exist a valid witness for the last statement (using the trapdoor as witness) to generate an accepting argument. Further, since Invariant W is assumed to not hold, MIM doesn't commit to a value w in either of the first two non-malleable commitments such that $R(x, w) = 1$. Therefore, there doesn't a valid witness for the first two statements either to generate an accepting argument. However, MIM produces an accepting argument. Thus, since $\mathcal{A}_{\mathsf{RWI}}$ exactly forwards the messages of MIM, with non-negligible probability p, $\mathcal{A}_{\mathsf{RWI}}$ has produces an accepting argument for a false statement breaking the Soundness of RWI which is a contradiction. Therefore, Invariant W holds in Hyb_0. □

Claim 31. *Invariant T holds in* Hyb_1.

Proof. Since there is no difference in the main thread between Hyb_0 and Hyb_1, the invariant continues to hold true. □

Claim 32. *Invariant W holds in* Hyb_1.

Proof. Since there is no difference in the main thread between Hyb_0 and Hyb_1, the invariant continues to hold true. □

Claim 33. *Assuming the existence of the trapdoor extractor* TDExt *for the trapdoor generation protocol* TDGen*, hiding of the non-malleable commitment scheme* NMCom *and the witness indistinguishability property of the scheme* RWI*,* Hyb_0 *is computationally indistinguishable from* Hyb_1 *except with probability atmost* $\frac{\epsilon}{10}$.

Proof. First, let's non-uniformly fix a first round message msg from $\mathsf{Sim}_{\mathsf{Hyb}}$. That is, this is the first round message which maximizes the adversary's probability of success.

Case 1: In Hyb_1, $q_{\mathsf{msg}} < \frac{\epsilon}{2}$.
That is, the probability that the adversary doesn't abort, conditioned on the first message is lesser than $\frac{\epsilon}{2}$. Then in this case, by applying the Chernoff bound, the number of non-aborting executions in the first step of Hyb_1 is lesser than λ except with 2^{ϵ} probability in which case both hybrids look identical as they just run the honest prover's algorithm and stop at the end of the protocol.

Case 2: suppose $2 \cdot \epsilon > q_{\mathsf{msg}} > \frac{\epsilon}{2}$
This is handled using case 1 and case 3 the same way as done in Section 5.4.2.

Case 3: suppose $q_{\mathsf{msg}} > 2 \cdot \epsilon$
Then, by the Chernoff bound, except with $2^{-\lambda}$ probability, in this case, the number of non-aborting transcripts is larger than λ and so $\mathsf{Sim}_{\mathsf{Hyb}}$ proceeds to the next step in Hyb_1. In that case, the only difference between the two hybrids is that in the left session, the simulator outputs "Special Abort" in the input extraction phase in Hyb_1. To prove that the two hybrids are indistinguishable, we will now show that the $\mathsf{Pr}[$ $\mathsf{Sim}_{\mathsf{Hyb}}$ outputs "Special Abort" $] \leq \frac{\epsilon}{10}$ in Hyb_1. $\mathsf{Sim}_{\mathsf{Hyb}}$ outputs "Special Abort" only if the algorithm TDExt fails.

By the definition of the scheme TDGen, the algorithm TDExt is successful except with negligible probability ϵ if given as input $(\mathsf{td}_1, \{\mathsf{td}_2^i, \mathsf{td}_3^i\}_{i=1}^3)$ such that $\mathsf{TDOut}(\mathsf{td}_1, \mathsf{td}_2^i, \mathsf{td}_3^i) = 1$ for all i. That is, TDExt is successful except with negligible probability ϵ if given as input 3 executions of the protocol TDGen.

Recall that in the left session, $\Pr_{(x,w \leftarrow X,W)}[\mathsf{Valid}(\mathsf{msg}, \mathsf{Trans}(P_2(x,w,\mathsf{st}), \mathsf{MIM})) = 1 | (\mathsf{msg}, \cdot) \leftarrow P_1(1^\lambda)] = q_{\mathsf{msg}}$ where the probability is over the random choices of (x,w) and the coins of the parties P_2, MIM. However, since each look-ahead thread in the left session is not identical to the main thread, we first need to argue that each look-ahead thread is indistinguishable from the main thread in the left session. If that is true, then, in each look-ahead thread, $Pr[\mathsf{TDOut}(\mathsf{td}_1^{\mathsf{MIM}\to P}, \mathsf{td}_2^{P\to\mathsf{MIM}}, \mathsf{td}_3^{\mathsf{MIM}\to P}) = 1] = q_{\mathsf{msg}}$. Note that this condition holds even in each look-ahead thread because each look-ahead thread only performs an honest execution of the protocol. Recall that $q_{\mathsf{msg}} > \epsilon$. Therefore, in $(\frac{3}{\epsilon})$ expected number of threads, the malicious verifier outputs 3 correct executions of the trapdoor generation protocol. Hence, by using the Markov inequality, in $(\lambda \cdot \frac{1}{\epsilon})$ threads, the extraction is successful except with negligible probability.

Therefore, the only remaining part of the proof is to show that every look-ahead thread in the left session is computationally indistinguishable from the main thread in the left session. Consider a series of hybrids described below where $\mathsf{Hyb}_{int,0}$ denotes the main thread of execution in the left session and $\mathsf{Hyb}_{int,3}$ denotes a look-ahead in the left session. We then show that every pair of consecutive hybrids is computationally indistinguishable thus proving that $\mathsf{Hyb}_{int,0}$ is computationally indistinguishable from $\mathsf{Hyb}_{int,3}$ which completes the proof.

- $\mathsf{Hyb}_{int,0}$ **- Main Thread:** In this hybrid, consider a simulator ${\mathsf{Sim}_{\mathsf{Hyb}}}^{int}$ that plays the role of the honest prover in the left session. This is the main thread. Note that ${\mathsf{Sim}_{\mathsf{Hyb}}}^{int}$ continues to play the role of an honest verifier in the right session.

- $\mathsf{Hyb}_{int,1}$ **- Changing 2nd Commitment:** In this hybrid, ${\mathsf{Sim}_{\mathsf{Hyb}}}^{int}$ does the following: in round 3 of the left session, compute $\mathsf{c}_{b,3}^{P\to V} \leftarrow \mathsf{NMCom}_3(w, \mathsf{c}_{b,1}^{P\to V}, \mathsf{c}_{b,2}^{V\to P}; \mathsf{r}_b)$.

- $\mathsf{Hyb}_{int,2}$ **- Switching Bounded Rewinding Secure WI proofs:** In this hybrid, ${\mathsf{Sim}_{\mathsf{Hyb}}}^{int}$ does the following: in round 3 of the left session, compute $\mathsf{rwi}_3^{P\to V} \leftarrow \mathsf{RWI}_3(\mathsf{rwi}_1^{P\to V}, \mathsf{rwi}_2^{V\to P}, \mathsf{st}_{\mathsf{RWI}}, \mathsf{w}_{\mathsf{RWI}})$

for the statement $\mathsf{st}_{\mathsf{RWI}} = (x, \mathsf{c}_{a,1}^{P\to V}, \mathsf{c}_{a,2}^{V\to P}, \mathsf{c}_{a,3}^{P\to V}, \mathsf{c}_{b,1}^{P\to V}, \mathsf{c}_{b,2}^{V\to P}, \mathsf{c}_{b,3}^{P\to V}, \mathsf{c}_{t,1}^{P\to V}, \mathsf{c}_{t,2}^{V\to P}, \mathsf{c}_{t,3}^{P\to V}, \mathsf{td}_1^{V\to P}, \mathsf{td}_2^{P\to V}, \mathsf{td}_3^{V\to P})$ using witness $(w, \bot, \mathsf{r}_b, \bot, \bot)$.

- $\mathsf{Hyb}_{int,3}$ **- Changing 1st Commitment:** In this hybrid, ${\mathsf{Sim}_{\mathsf{Hyb}}}^{int}$ does the following: in round 3 of the left session, compute $\mathsf{c}_{a,3}^{P\to V} \leftarrow \mathsf{NMCom}_3(\bot, \mathsf{c}_{a,1}^{P\to V}, \mathsf{c}_{a,2}^{V\to P}; \mathsf{r}_a)$. This hybrid now corresponds to the look-ahead thread.

Lemma 5. *Assuming the hiding property of the non-malleable commitment scheme* NMCom*,* $\mathsf{Hyb}_{int,0}$ *is computationally indistinguishable from* $\mathsf{Hyb}_{int,1}$*.*

Proof. The only difference between $\mathsf{Hyb}_{int,0}$ and $\mathsf{Hyb}_{int,1}$ is that in $\mathsf{Hyb}_{int,1}$, the simulator now computes the second non-malleable commitment using input w in the left session. Suppose there exists an adversary V^* that can distinguish between the two hybrids with non-negligible probability. We will use V^* to design an adversary $\mathcal{A}_{\mathsf{Hid}}$ that breaks the hiding of the non-malleable commitment scheme.

$\mathcal{A}_{\mathsf{Hid}}$ interacts with a challenger $\mathcal{C}_{\mathsf{Hid}}$. $\mathcal{A}_{\mathsf{Hid}}$ performs the role of ${\mathsf{Sim}_{\mathsf{Hyb}}}^{int}$ in its interaction with V^* almost exactly as done in $\mathsf{Hyb}_{int,0}$. $\mathcal{A}_{\mathsf{Hid}}$ interacts with a challenger $\mathcal{C}_{\mathsf{Hid}}$ and receives a first round message which is set as $\mathsf{c}_{b,1}^{P\to V}$ in its interaction with V^* in round 1 of protocol $\pi^{\mathsf{SE-PZK}}$ in the left session. $\mathcal{A}_{\mathsf{Hid}}$ receives a value $\mathsf{c}_{b,2}^{V\to P}$ in round 2 of protocol $\pi^{\mathsf{SE-PZK}}$ in the left session which it sends to $\mathcal{C}_{\mathsf{Hid}}$ as its second round message. Then, $\mathcal{A}_{\mathsf{Hid}}$ sends the pair of values $(\bot, w)$ to $\mathcal{C}_{\mathsf{Hid}}$. Recall that NMCom is a delayed-input scheme. $\mathcal{A}_{\mathsf{Hid}}$ receives a third round message from $\mathcal{C}_{\mathsf{Hid}}$ which is either a commitment to $\bot$ or w. This is sent to V^* as the value $\mathsf{c}_{b,3}^{P\to V}$ in round 3 in the left session. The rest of the protocol is performed exactly as in $\mathsf{Hyb}_{int,0}$.

Observe that the first case corresponds to $\mathsf{Hyb}_{int,0}$ while the second case corresponds to $\mathsf{Hyb}_{int,1}$. Therefore, if the adversary V^* can distinguish between these two hybrids with non-negligible probability, $\mathcal{A}_{\mathsf{Hid}}$ will use the same guess to break the hiding of the non-malleable commitment scheme NMCom with non-negligible probability which is a contradiction. □

Claim 34. *Assuming the witness indistinguishability property of the scheme* RWI*,* $\mathsf{Hyb}_{int,1}$ *is computationally indistinguishable from* $\mathsf{Hyb}_{int,2}$*.*

Proof. **Note:** We don't need the bounded rewinding property of the WI scheme in this proof. That is, here, we rely only on standard witness indistinguishability.

The only difference between $\mathsf{Hyb}_{int,1}$ and $\mathsf{Hyb}_{int,2}$ is that in $\mathsf{Hyb}_{int,2}$, the simulator now computes the bounded rewinding secure WI proof using a different witness in the left session. Suppose there exists an adversary V^* that can distinguish between the two hybrids with non-negligible probability. We will use V^* to design an adversary $\mathcal{A}_{\mathsf{RWI}}$ that breaks the security of the bounded rewinding secure WI scheme with non-negligible probability.

$\mathcal{A}_{\mathsf{RWI}}$ interacts with a challenger $\mathcal{C}_{\mathsf{RWI}}$. $\mathcal{A}_{\mathsf{RWI}}$ performs the role of $\mathsf{Sim}_{\mathsf{Hyb}}$ in its interaction with V^* almost exactly as done in $\mathsf{Hyb}_{int,1}$. From $\mathcal{C}_{\mathsf{RWI}}$, $\mathcal{A}_{\mathsf{RWI}}$ receives a first round message rwi_1 which is set as $\mathsf{rwi}_1^{P\to V}$ in its interaction with V^* in round 1 of protocol $\pi^{\mathsf{SE-PZK}}$ in the left session. On receiving $\mathsf{rwi}_2^{V\to P}$ in round 2 in the left session, $\mathcal{A}_{\mathsf{RWI}}$ forwards this to $\mathcal{C}_{\mathsf{RWI}}$ as its second round message. Then, $\mathcal{A}_{\mathsf{RWI}}$ sends the statement $\mathsf{st}_{\mathsf{RWI}} = (x, \mathsf{c}_{a,1}^{P\to V}, \mathsf{c}_{a,2}^{V\to P}, \mathsf{c}_{a,3}^{P\to V}, \mathsf{c}_{b,1}^{P\to V}, \mathsf{c}_{b,2}^{V\to P}, \mathsf{c}_{b,3}^{P\to V}, \mathsf{c}_{t,1}^{P\to V}, \mathsf{c}_{t,2}^{V\to P}, \mathsf{c}_{t,3}^{P\to V}, \mathsf{td}_1^{V\to P}, \mathsf{td}_2^{P\to V}, \mathsf{td}_3^{V\to P})$ to $\mathcal{C}_{\mathsf{RWI}}$ where the other values are generated as in $\mathsf{Hyb}_{int,1}$. $\mathcal{A}_{\mathsf{RWI}}$ also sends the pair of witnesses $(w, \mathsf{r}_a, \bot, \bot, \bot)$ and $(w, \bot, \mathsf{r}_b, \bot, \bot)$. Recall that RWI is a delayed-input scheme. $\mathcal{A}_{\mathsf{RWI}}$ receives a third round message rwi_3 which is set as $\mathsf{rwi}_3^{P\to V}$ in its interaction with V^* in round 3 of protocol $\pi^{\mathsf{SE-PZK}}$ in the left session. The rest of protocol $\pi^{\mathsf{SE-PZK}}$ is performed exactly as in $\mathsf{Hyb}_{int,1}$.

Observe that the first case corresponds to $\mathsf{Hyb}_{int,1}$ while the second case corresponds to $\mathsf{Hyb}_{int,2}$. Therefore, if the adversary V^* can distinguish between these two hybrids with non-negligible probability, $\mathcal{A}_{\mathsf{RWI}}$ will use the same guess to break the witness indistinguishability property of the scheme RWI with non-negligible probability which is a contradiction. □

Lemma 6. *Assuming the hiding property of the non-malleable commitment scheme* NMCom, $\mathsf{Hyb}_{int,2}$ *is computationally indistinguishable from* $\mathsf{Hyb}_{int,3}$.

Proof. This proof is identical to the proof of Lemma 5 described above. □

This completes the proof of Claim 33.

□

Claim 35. *Assuming* NMCom *is a secure non-malleable commitment scheme, Invariant T holds in* Hyb_2.

Proof. We know that Invariant T holds in Hyb_1. The only difference between Hyb_1 and Hyb_2 is that in Hyb_2, the simulator now computes the non-malleable commitment in the main thread of the left session using the adversary's trapdoor value. Assume for the sake of contradiction that Invariant T doesn't hold in Hyb_2. That is, there exists an adversary MIM such that it causes event E_t to occur with non-negligible probability. We will use $\mathcal{A}$ to design an adversary $\mathcal{A}_{\mathsf{NMCom}}$ that breaks the security of the non-malleable commitment scheme.

$\mathcal{A}_{\mathsf{NMCom}}$ interacts with a challenger $\mathcal{C}_{\mathsf{NMCom}}$. $\mathcal{A}_{\mathsf{NMCom}}$ performs the role of $\mathsf{Sim}_{\mathsf{Hyb}}$ in its interaction with MIM exactly as done in Hyb_1. $\mathcal{A}_{\mathsf{NMCom}}$ receives a first round message c_1^L on the left side from $\mathcal{C}_{\mathsf{NMCom}}$ which is set as $\mathsf{c}_1^{P\to V}$ in its interaction with $\mathcal{A}$ in round 1 of protocol $\pi^{\mathsf{SE-PZK}}$ in the left session. On receiving $\mathsf{c}_1^{P\to V}$ from MIM in the right session, $\mathcal{A}_{\mathsf{NMCom}}$ forwards this to $\mathcal{C}_{\mathsf{NMCom}}$ as its first round message on the right side.

$\mathcal{A}_{\mathsf{NMCom}}$ receives a value c_2^R from $\mathcal{C}_{\mathsf{NMCom}}$ as the second round message on the right side which it sets as the value $\mathsf{c}_2^{V\to P}$ in the right session in round 2 of protocol $\pi^{\mathsf{SE-PZK}}$ on the main thread. Then, on receiving $\mathsf{c}_{t,2}^{V\to P}$ in the main thread in the left session, $\mathcal{A}_{\mathsf{NMCom}}$ sends this to $\mathcal{C}_{\mathsf{NMCom}}$ as its second round message on the left side. $\mathcal{A}_{\mathsf{NMCom}}$ generates the messages in the other look-ahead threads on its own. Then, after the input extraction phase, $\mathcal{A}_{\mathsf{NMCom}}$ sends the pair of values $(\perp, \mathsf{t}_V)$ to $\mathcal{C}_{\mathsf{NMCom}}$. Recall that NMCom is a delayed-input scheme.

$\mathcal{A}_{\mathsf{NMCom}}$ receives a third round message c_3^L on the left side from $\mathcal{C}_{\mathsf{NMCom}}$ which is either a commitment to $\perp$ or t_V. This is sent to MIM as the value $\mathsf{c}_3^{P\to V}$ in the main thread on the left session. The rest of protocol $\pi^{\mathsf{SE-PZK}}$ is performed exactly as in Hyb_1. On receiving the value $\mathsf{c}_3^{P\to V}$ from MIM in the main thread on the right session, $\mathcal{A}_{\mathsf{NMCom}}$ sends it to $\mathcal{C}_{\mathsf{NMCom}}$ as its third round message in the right side. Note that in all the other look-ahead threads, in the left session, $\mathcal{A}_{\mathsf{NMCom}}$ continues to compute the non-malleable commitment using input $\perp$ as done in Hyb_1. Recall that $\mathsf{NMCom}_3(\perp)$ is indistinguishable from a random string and so $\mathcal{A}_{\mathsf{NMCom}}$ can generate this by picking a uniformly random string without knowing the randomness used to generate $\mathsf{c}_1^{P\to V}$.

Observe that the first case corresponds to Hyb_1 while the second case corresponds to Hyb_2. Now, let's analyze why $\mathcal{A}_{\mathsf{NMCom}}$ breaks the security of the scheme NMCom. We know that event E_t doesn't hold in Hyb_2. That is, the adversary MIM, in the right session of the main thread, using the non-malleable commitment, commits to a valid trapdoor t^* for the trapdoor generation messages of $\mathsf{Sim}_{\mathsf{Hyb}}$ with non-negligible probability ϵ. Recall that this is same as the messages sent by $\mathcal{A}_{\mathsf{NMCom}}$ to $\mathcal{C}_{\mathsf{NMCom}}$ as its commitment messages on the right side. In Hyb_1, since the invariant holds, the adversary did not commit to a valid trapdoor.

Therefore, when the value committed to by the honest party in the left execution changes, the value committed to by the adversary in the right execution has also changed except with negligible

probability. This breaks the security of the scheme NMCom which is a contradiction. Thus, the invariant must hold in Hyb_2 as well. □

Claim 36. *Assuming the soundness of the bounded rewinding secure WI argument system, Invariant W holds in* Hyb_2.

Proof. This is identical to the proof of Claim 30. □

Claim 37. *Assuming the hiding property of the non-malleable commitment scheme* NMCom, Hyb_1 *is computationally indistinguishable from* Hyb_2 *except with probability at most* $\frac{\epsilon}{10}$.

Proof. The only difference between Hyb_1 and Hyb_2 is that in Hyb_2, the simulator now computes the non-malleable commitment using the adversary's trapdoor value in the left session. Suppose there exists an adversary V^* that can distinguish between the two hybrids with non-negligible probability greater than $\frac{\epsilon}{10}$, we can use V^* to design an adversary $\mathcal{A}_{\mathsf{Hid}}$ that breaks the hiding of the non-malleable commitment scheme. The rest of the proof is similar to the proof of Claim 35. □

Claim 38. *Assuming* RWI *is a bounded rewinding secure witness indistinguishable argument and the existence of an extractor* $\mathsf{Ext}_{\mathsf{NMCom}}$ *for the non-malleable commitment scheme* NMCom, *Invariant T holds in* Hyb_3.

Proof. We know that the invariant holds in Hyb_2. The only difference between Hyb_2 and Hyb_3 is that in Hyb_3, in the main thread of the left session, the simulator now computes the bounded rewinding secure WI proof using a different witness. Assume for the sake of contradiction that Invariant T doesn't hold in Hyb_3. That is, there exists an adversary MIM such that that causes event E_t to occur with non-negligible probability.

We will use $\mathcal{A}$ to design an adversary $\mathcal{A}_{\mathsf{RWI}}$ that breaks the security of the bounded rewinding secure WI scheme.

$\mathcal{A}_{\mathsf{RWI}}$ interacts with a challenger $\mathcal{C}_{\mathsf{RWI}}$. $\mathcal{A}_{\mathsf{RWI}}$ performs the role of $\mathsf{Sim}_{\mathsf{Hyb}}$ in its interaction with MIM exactly as done in Hyb_2. From $\mathcal{C}_{\mathsf{RWI}}$, $\mathcal{A}_{\mathsf{RWI}}$ receives a first round message rwi_1 which is set as $\mathsf{rwi}_1^{P\to V}$ in its interaction with MIM in round 1 of protocol $\pi^{\mathsf{SE-PZK}}$ in the left session. Then, $\mathcal{A}_{\mathsf{RWI}}$ creates a set of L look ahead threads (L being the parameter for the protocol RWI which is set to 5 here). For each thread, on receiving $\mathsf{rwi}_2^{V\to P}$ in round 2 from MIM in the left session, $\mathcal{A}_{\mathsf{RWI}}$ forwards this to $\mathcal{C}_{\mathsf{RWI}}$ as its second round message. For each thread, $\mathcal{A}_{\mathsf{RWI}}$ also sends the statement $\mathsf{st}_{\mathsf{RWI}} = (x, \mathsf{c}_{a,1}^{P\to V}, \mathsf{c}_{a,2}^{V\to P}, \mathsf{c}_{a,3}^{P\to V}, \mathsf{c}_{b,1}^{P\to V}, \mathsf{c}_{b,2}^{V\to P}, \mathsf{c}_{b,3}^{P\to V}, \mathsf{c}_{t,1}^{P\to V}, \mathsf{c}_{t,2}^{V\to P}, \mathsf{c}_{t,3}^{P\to V}, \mathsf{td}_1^{V\to P}, \mathsf{td}_2^{P\to V}, \mathsf{td}_3^{V\to P})$ where the other values are generated as in Hyb_2.

In the main thread, $\mathcal{A}_{\mathsf{RWI}}$ also sends the pair of witnesses $(w, \mathsf{r}_a, \bot, \bot, \bot)$ and $(\bot, \bot, \bot, \mathsf{t}_V, \mathsf{r}_t)$ where t_V is computed in the input extraction phase. For each look-ahead thread, $\mathcal{A}_{\mathsf{RWI}}$ sends the witness $(w, \bot, \mathsf{r}_b, \bot, \bot)$. Recall that RWI is a delayed-input scheme. For each thread, $\mathcal{A}_{\mathsf{RWI}}$ receives a third round message rwi_3 which is set as $\mathsf{rwi}_3^{P\to V}$ in its interaction with MIM in round 3 of protocol $\pi^{\mathsf{SE-PZK}}$ in the left session. The rest of protocol $\pi^{\mathsf{SE-PZK}}$ is performed exactly as in Hyb_2. Observe that on each thread, since V^* produces a non-aborting transcript with non-negligible probability, $\mathcal{A}_{\mathsf{RWI}}$ (via the trapdoor extractor TDExt) extracts a valid trapdoor with non-negligible probability.

Observe that the first case corresponds to Hyb_2 while the second case corresponds to Hyb_3.

Now, let's see how $\mathcal{A}_{\mathsf{RWI}}$ breaks the bounded rewinding security of the WI scheme. $\mathcal{A}_{\mathsf{RWI}}$ runs the extractor $\mathsf{Ext}_{\mathsf{NMCom}}$ of the non-malleable commitment scheme using the messages in all the threads in the right session that correspond to the non-malleable commitment from MIM. Let the output of $\mathsf{Ext}_{\mathsf{NMCom}}$ be t^*. If $\mathsf{TDValid}(\mathsf{td}_1^{V\to P}, \mathsf{t}^*) = 1$, where $\mathsf{td}_1^{V\to P}$ is the message sent by $\mathsf{Sim}_{\mathsf{Hyb}}$

in the right session, then $\mathcal{A}_{\mathsf{RWI}}$ outputs case 2 indicating that the bounded rewinding secure WI was constructed using the second witness. Else, it outputs case 1.

Let's analyze why this works. We know that event E_t doesn't hold in Hyb_3. That is, the adversary MIM, using the non-malleable commitment, commits to a valid trapdoor t^* for the trapdoor generation messages of $\mathsf{Sim}_{\mathsf{Hyb}}$ in the right session with non-negligible probability ϵ. Therefore, since the invariant holds in Hyb_2 with non-negligible probability, when the extractor $\mathsf{Ext}_{\mathsf{NMCom}}$ outputs a valid trapdoor t^*, it must be the case that we are in Hyb_3 with non-negligible probability. That is, when $\mathsf{Ext}_{\mathsf{NMCom}}$ outputs a valid trapdoor, it corresponds to $\mathcal{A}_{\mathsf{RWI}}$ receiving a proof using the second witness and otherwise, it corresponds to $\mathcal{A}_{\mathsf{RWI}}$ receiving a proof using the first witness. Thus, $\mathcal{A}_{\mathsf{RWI}}$ breaks the security of the scheme RWI which is a contradiction. Hence, the invariant holds in Hyb_3 as well. □

Claim 39. *Assuming the soundness of the bounded rewinding secure WI argument system, Invariant W holds in* Hyb_3.

Proof. This is identical to the proof of Claim 30. □

Claim 40. *Assuming* RWI *is a bounded rewinding secure witness indistinguishable argument,* Hyb_2 *is computationally indistinguishable from* Hyb_3 *except with probability at most* $\frac{\epsilon}{10}$.

Proof. The only difference between Hyb_2 and Hyb_3 is that in Hyb_3, in the main thread in the left session, the simulator now computes the bounded rewinding secure WI proof using a different witness. Suppose there exists an adversary V^* that can distinguish between the two hybrids with non-negligible probability greater than $\frac{\epsilon}{10}$. We can use V^* to design an adversary $\mathcal{A}_{\mathsf{RWI}}$ that breaks the bounded rewinding security of the WI scheme with non-negligible probability. The rest of the proof is similar to the proof of Claim 38.

□

Claim 41. *Assuming* NMCom *is a secure non-malleable commitment scheme, Invariant T holds in* Hyb_4.

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

Claim 42. *Assuming the soundness of the bounded rewinding secure WI argument system, Invariant W holds in* Hyb_4.

Proof. This is identical to the proof of Claim 30. □

Claim 43. *Assuming the hiding property of the non-malleable commitment scheme* NMCom, Hyb_3 *is computationally indistinguishable from* Hyb_4 *except with probability at most* $\frac{\epsilon}{10}$.

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

Claim 44. *Invariant T holds in* Hyb_5.

Proof. Since Hyb_4 and Hyb_5 are identical, the invariant continues to hold true. □

Claim 45. *Invariant W holds in* Hyb_5.

Proof. Since Hyb_4 and Hyb_5 are identical, the invariant continues to hold true. □

Claim 46. *Assuming the existence of the trapdoor extractor* TDExt *for the trapdoor generation protocol* TDGen, Hyb_4 *is computationally indistinguishable from* Hyb_5 *except with probability at most* $\frac{\epsilon}{10}$.

Proof. This is similar to the proof of Claim 6 in Section 5.4.2. □

This completes the proof of the Simulation Security property.

Remark: In the entire proof, note that it is actually sufficient to use an extractable commitment scheme secure against 1-rewinding in place of each of the first two non-malleable commitments - that is, the ones used to commit to the actual witness for the instance.

7.2.4 Simulation Extractability

In this section, we will prove that the Simulation Extractability property holds in Hyb_5 - the ideal world.

The description of the extractor E is given in Figure 9.

The extractor E does the following:

1. Run the extractor $\mathsf{Ext}_{\mathsf{NMCom}}$ of the scheme NMCom using the non-malleable commitment messages in the right session for index a and compute w_1. Note that $\mathsf{Ext}_{\mathsf{NMCom}}$ internally rewinds MIM several (polynomially many) times on the right session to extract the value.
2. Similarly, compute $w_2 = \mathsf{Ext}_{\mathsf{NMCom}}(\mathsf{c}_{b,1}^{P\to V}, \{\mathsf{c}_{b,2}^{V\to P}, \mathsf{c}_{b,3}^{P\to V}\})$.
3. If $R(x^*, w_1) = 1$, output w_1.
4. Else, if $R(x^*, w_2) = 1$, output w_2.
5. Else, output Abort.

Figure 9: Description of Extractor

We now show that

$$\Pr\left[E^{\mathsf{MIM}}(z, \mathsf{Trans}) = w \wedge R(x^*, w) = 1 \;\middle|\; \langle \mathsf{Sim}_{\mathsf{SE}}(x, z), \mathsf{MIM}(z)\rangle_R = 1\right] \geq (1 - \mathsf{negl}(\lambda)),$$

where x is the statement used by $\mathsf{Sim}_{\mathsf{SE}}$ in the left session and x^* is the statement used by MIM in the right session.

Now, in the ideal world Hyb_5, (which corresponds to the interaction of MIM with $\mathsf{Sim}_{\mathsf{SE}}$), suppose $\mathsf{Sim}_{\mathsf{SE}}(x, z), \mathsf{MIM}(z)\rangle_R = 1$. That is, suppose MIM produces an accepting argument on the right session. Since Invariant W holds, we know that, using either the first or the second non-malleable commitment, MIM commits to a valid witness w such that $R(x^*, w) = 1$. Therefore, since the extractor of the non-malleable commitment scheme $\mathsf{Ext}_{\mathsf{NMCom}}$ is successful except with non-negligible probability given the transcript of all the executions, the extractor E outputs a witness w such that $R(x^*, w) = 1$ successfully except with negligible probability. This completes the proof of the Simulation Extractability property.

Here, note that though the extractor $\mathsf{Ext}_{\mathsf{NMCom}}$ might over-extract, that doesn't affect our proof. That is, since the invariant W holds, we know that the non-malleable commitment is indeed well-formed and so the extractor doesn't over-extract.

8 Three Round List Coin Tossing

We first define the notion of list coin tossing.

Definition 10. *An n party protocol π in the simultaneous message setting is a list coin tossing protocol if for every PPT adversary $\mathcal{A}$ corrupting at most $(n1)$ parties, there exists an expected PPT simulator $\mathcal{S}$ and a polynomial p such that the output of the experiments* REAL *and* IDEAL *defined below are indistinguishable. In the real world, we denote the result of running protocol π with adversary $\mathcal{A}$ as a pair $(\mathsf{c}, \mathsf{view}_{\mathcal{A}})$ where $\mathsf{c} \in \{0,1\}^l \cup \{\perp\}$ is the output of the protocol and $\mathsf{view}_{\mathcal{A}}$ is the view of the adversary $\mathcal{A}$. Similarly, we use the pair $(\tilde{\mathsf{c}}, \mathsf{view}_{\mathcal{S}})$ in the ideal world.*

We use l to denote the output length of the protocol.

$\mathsf{REAL}(1^\lambda, 1^\mathsf{l})$	$\mathsf{IDEAL}(1^\lambda, 1^\mathsf{l})$
$(\mathsf{c}, \mathsf{view}_{\mathcal{A}}) \leftarrow \mathsf{REAL}_{\pi,\mathcal{A}}(1^\lambda, 1^l)$	$(\mathsf{c}_1, \ldots, \mathsf{c}_{p(\lambda)}) \leftarrow \{0,1\}^{l \cdot p(\lambda)}$ where p is some polynomial.
	$(\tilde{\mathsf{c}}, \mathsf{view}_{\mathcal{S}}) \leftarrow \mathcal{S}^{\mathcal{A}}(\mathsf{c}_1, \ldots, \mathsf{c}_{p(\lambda)}, 1^\lambda, 1^l)$
Output $(\mathsf{c}, \mathsf{view}_{\mathcal{A}})$	If $\tilde{\mathsf{c}} \in \{\mathsf{c}_1, \ldots, \mathsf{c}_{p(\lambda)}\}$, then output $(\tilde{\mathsf{c}}, \mathsf{view}_{\mathcal{S}})$
	Else, output fail.

Table 1: List Coin Tossing

8.1 Construction

Consider n parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ who wish to evaluate the List Coin Tossing functionality. In this section, we construct a protocol for computing the List Coin Tossing functionality using the Simulation Extractable Promise ZK argument system from the previous section. Formally, we prove the following theorem:

Theorem 10. *Assuming the existence of polynomially secure injective one way functions, the protocol π^{CT} presented below is a 3 round protocol for the list Coin Tossing functionality in the plain model secure against any adversary that corrupts up to $(n-1)$ parties.*

As mentioned earlier, we note that by applying the transformation[10] of [GMPP16] on the protocol from Theorem 3 for the two-party case, we can obtain a four round two-party list coin-tossing protocol in the *unidirectional-message* model. This result overcomes the barrier of five rounds for standard two-party coin-tossing established by [KO04].

Corollary 11 (Informal)**.** *Assuming the existence of polynomially secure injective one-way functions, there exists a four round two-party list coin-tossing protocol in the unidirectional-message model with black-box simulation.*

We first list some notation and the primitives used before describing the construction.

Primitives Used.

- Let Com be any non-interactive commitment scheme. We know that such a commitment scheme can be built assuming injective one way functions.

[10]The work of Garg et al. [GMPP16] establishes an impossibility result for three round multiparty coin-tossing by transforming any three round two-party coin-tossing protocol in the simultaneous-message model into a four round two-party coin-tossing protocol in the unidirectional-message model, and then invoking the impossibility of [KO04].

- $\pi^{\mathsf{SE-PZK}} = (\mathsf{Prove}, \mathsf{Verify}, \mathsf{Valid})$ is the 3 round Simulation Extractable Promise ZK argument system from Section 7. Recall that this protocol has a delayed-input property. Let E denote the polynomial time extractor of this protocol as constructed in Section 7.2.

  For all $i \in [3]$, let's use Prove_i to denote the algorithm used by the prover P and Verify_i to denote the algorithm used by the verifier V to compute the i^{th} round messages. Further, let Verify_4 denote the algorithm used by the verifier V to compute the output bit 0 or 1.

NP Languages. In our construction, we use proofs for NP language L characterized by the following relation R.
Statement : $\mathsf{st} = (\mathsf{c}, \mathsf{r})$
Witness : $\mathsf{w} = (\mathsf{s})$
$R(\mathsf{st}, \mathsf{w}) = 1$ if and only if $\mathsf{c} = \mathsf{Com}(\mathsf{r}; \mathsf{s})$.

Notation :

- We assume broadcast channels.
- λ denotes the security parameter.
- In the superscript, we use $i \to j$ to denote that the message was sent by party P_i with intended recipient as party P_j. (recall that all messages are broadcast).
- The round number of the sub-protocol $\pi^{\mathsf{SE-PZK}}$ being used is written in the subscript.

The protocol π^{CT} for coin tossing in three rounds is described in Figure 10.

8.2 Security Proof

In this section, we formally prove Theorem 10.

Before that, we list 2 crucial properties of the simulation extractable promise ZK argument system from Section 7 that we use in the proof:

- Recall that the first round message of the simulator is identical to the first round message of an honest party. That is, $\mathsf{Sim}_1(z) = P_1(1^\lambda)$.
- Recall that the simulator Sim_2 has to rewind and create a set of look-ahead threads. However, for a fixed first round message sent by Sim_1, using just 1 set of look-ahead threads, the simulator can create several simulated arguments.

Consider an adversary $\mathcal{A}$ who corrupts t parties where $t < n$. Let $\mathsf{Sim}_{\mathsf{SE}}$ denote the simulator of the Simulation Extractable Promise ZK system as defined in Section 7.2.

8.2.1 Description of Simulator

We first informally discuss the strategy followed by the simulator Sim. First, Sim runs the protocol to completion honestly to check if the adversary $\mathcal{A}$ aborts. If so, Sim aborts. If not, Sim now knows that the probability that $\mathcal{A}$ doesn't abort is at least $\frac{1}{\mathsf{poly}(\lambda)}$ for some polynomial. Sim now estimates this probability p by rewinding the adversary several times and behaving honestly in each rewound thread as done in [GK96, Lin17].

After estimating p, Sim rewinds the adversary back to the beginning and performs a fresh execution but this time it produces simulated arguments for the protocol $\pi^{\mathsf{SE-PZK}}$ and additionally

1. **Round 1:** $\forall j \in [n]$ with $j \neq i$, P_i does the following:
   - Generate $(\mathsf{prove}_1^{i\to j}) \leftarrow \mathsf{Prove}_1(1^\lambda)$ and $(\mathsf{ver}_1^{i\to j}) \leftarrow \mathsf{Verify}_1(1^\lambda)$.
   - Broadcast $(\mathsf{prove}_1^{i\to j}, \mathsf{ver}_1^{i\to j})$.
2. **Round 2:** P_i does the following:
   - For each $j \in [n]$ with $j \neq i$, do:
     - Let $\tau_1^{i\to j}$ denote the transcript of protocol $\pi^{\mathsf{SE-PZK}}$ after round 1 with prover P_i and verifier P_j. That is, $\tau_1^{i\to j} = (\mathsf{prove}_1^{i\to j}, \mathsf{ver}_1^{j\to i})$.
     - Similarly, let $\tau_1^{j\to i}$ denote the transcript of protocol $\pi^{\mathsf{SE-PZK}}$ after round 1 with prover P_j and verifier P_i.
     - Generate $(\mathsf{prove}_2^{i\to j}) \leftarrow \mathsf{Prove}_2(\tau_1^{i\to j})$ and $(\mathsf{ver}_2^{i\to j}) \leftarrow \mathsf{Verify}_2(\tau_1^{j\to i})$.
   - Compute $\mathsf{c}_i = \mathsf{Com}(\mathsf{r}_i; \mathsf{s}_i)$ using random strings $(\mathsf{r}_i, \mathsf{s}_i)$. Here, the length of r_i is l which is the output length of the coin tossing protocol.
   - Broadcast $(\mathsf{c}_i, \{\mathsf{prove}_2^{i\to j}, \mathsf{ver}_2^{i\to j}\}_{j\neq i})$.
3. **Round 3:** P_i does the following:
   - For each $j \in [n]$ with $j \neq i$, do:
     - Let $\tau_2^{i\to j}$ denote the transcript of protocol $\pi^{\mathsf{SE-PZK}}$ after round 2 with prover P_i and verifier P_j.
     - Similarly, let $\tau_2^{j\to i}$ denote the transcript of protocol $\pi^{\mathsf{SE-PZK}}$ after round 2 with prover P_j and verifier P_i.
     - Generate $(\mathsf{prove}_3^{i\to j}) \leftarrow \mathsf{Prove}_3(\tau_2^{i\to j}, \mathsf{st}^{i\to j}, \mathsf{w}^{i\to j})$ for the statement $\mathsf{st}^{i\to j} = (\mathsf{c}_i, \mathsf{r}_i) \in L$ using witness $\mathsf{w}^{i\to j} = \mathsf{s}_i$.
     - Generate $(\mathsf{ver}_3^{i\to j}) \leftarrow \mathsf{Verify}_3(\tau_2^{j\to i})$.
   - Broadcast $(\mathsf{r}_i, \{\mathsf{prove}_3^{i\to j}, \mathsf{ver}_3^{i\to j}\}_{j\neq i})$.
4. **Output Computation:** P_i does the following:
   - For each $j \in [n]$ with $j \neq i$, do:
     - Let $\tau_3^{j\to i}$ denote the transcript of protocol $\pi^{\mathsf{SE-PZK}}$ after round 3 with prover P_j and verifier P_i.
     - Abort if $\mathsf{Verify}_4(\tau_3^{j\to i}, \mathsf{st}^{j\to i}) \neq 1$ where $\mathsf{st}^{j\to i} = (\mathsf{c}_j, \mathsf{r}_j)$. In particular, send a global abort signal to all parties so that everyone aborts.
   - Else, compute output $\mathsf{y}_i = \bigoplus_{i\in[n]} \mathsf{r}_i$.

Figure 10: 3 round malicious secure protocol π^{CT} for list coin tossing.

doesn't open the values r_i correctly. If the adversary aborts, Sim rewinds back to the beginning again and repeats a fresh execution until the adversary doesn't abort. Since the probability of not aborting is p, we know that this step takes an expected number of $\frac{1}{p}$ executions. Essentially, the point of this step is to extract the adversary's r values.

After this, Sim rewinds $\mathcal{A}$ back to the end of the 2nd round. Now, for each output c from the ideal functionality, Sim tries to force this value in the 3rd round. Once again, notice that $\mathcal{A}$ may abort. Sim rewinds and repeats this round using a fresh output from the ideal functionality each time till the adversary doesn't abort. We know that this step too takes an expected number of $\frac{1}{p}$

executions.

The strategy of the simulator Sim against a malicious adversary $\mathcal{A}$ now follows:

Step 1 - Check Abort:

1. For each honest party P_i, Sim runs an honest execution of the protocol π^{CT} with $\mathcal{A}$.
2. Sim outputs Abort if the protocol aborts. Else, proceeds to the next step.

Step 2 - Abort Probability Estimation:

1. Sim now rewinds back to the end of round 1 of the protocol. Then, Sim creates a set of look-ahead threads that run only rounds 2 and 3 of the protocol honestly exactly as done in Step 1.
2. The above procedure happens till Sim receives non-aborting transcripts in $(12 \cdot \lambda)$ threads. Let the total number of look-ahead threads created to achieve this be T.
3. Set $\epsilon' = \frac{12\lambda}{T}$ as the probability with which the adversary doesn't abort.

Step 3 - Query to Ideal Functionality: Sim does the following:

1. Set $K = \frac{\lambda^2}{\epsilon'}$.
2. If $K \geq 2^\lambda$, Abort.
3. Else, output K to the ideal functionality and receives a set of outputs $(\mathsf{ans}_1, \ldots \mathsf{ans}_K)$.

Step 4 - Learning Adversary's Randomness:
Sim rewinds $\mathcal{A}$ back to the end of round 1 of the protocol and does the following. Note that this step also involves rewinding as elaborated below. Further, here we crucially use the two properties of the simulation extractable promise ZK argument mentioned at the beginning of the section. First, Sim sets the counter value as 0. Also, Sim invokes the simulator Sim_2 of $\pi^{\mathsf{SE-PZK}}$ with the probability of not abort - $p = \frac{\epsilon'}{\lambda}$. We denote the simulator Sim_2 of the argument $\pi^{\mathsf{SE-PZK}}$ by $\mathsf{Sim}_{\mathsf{SE}}$.

1. **Round 2:** For every honest party P_i:
   - For each $j \in [n]$ with $j \neq i$, do:
     - Let τ_1 denote the protocol transcript so far.
     - Generate $(\mathsf{prove}_2^{i \to j}) \leftarrow \mathsf{Sim}_{\mathsf{SE}}(\tau_1)$ and $(\mathsf{ver}_2^{i \to j}) \leftarrow \mathsf{Sim}_{\mathsf{SE}}(\tau_1)$.
   - Compute $\mathsf{c}_i = \mathsf{Com}(0; \mathsf{s}_i)$ using a random string (s_i). Also, pick a random string r_i to be used in round 3.
   - Broadcast $(\mathsf{c}_i, \{\mathsf{prove}_2^{i \to j}, \mathsf{wi}_2^{i \to j}\}_{j \neq i})$.
2. **Round 3:** For every honest party P_i, do the following:
   - For each $j \in [n]$ with $j \neq i$, do:
     - Let τ_2 denote the protocol transcript so far.
     - Generate $(\mathsf{prove}_3^{i \to j}) \leftarrow \mathsf{Sim}_{\mathsf{SE}}(\tau)$ for the statement $\mathsf{st}^{i \to j} = (\mathsf{c}_i, \mathsf{r}_i) \in L$.
     - Generate $(\mathsf{ver}_3^{i \to j}) \leftarrow \mathsf{Sim}_{\mathsf{SE}}(\tau_2)$.
   - Broadcast $(\mathsf{r}_i, \{\mathsf{prove}_3^{i \to j}, \mathsf{wi}_3^{i \to j}\}_{j \neq i})$.
3. **Output Computation:** Sim does the following:

- For every honest party P_i and each $j \in [n]$ with $j \neq i$, do:
  - Let $\tau_3^{j \to i}$ denote the transcript of protocol $\pi^{\mathsf{SE-PZK}}$ after round 3 with prover P_j and verifier P_i.
  - If $\mathsf{Verify}_4(\tau_3^{j \to i}, \mathsf{st}^{j \to i}) = 1$ where $\mathsf{st}^{j \to i} = (\mathsf{c}_j, \mathsf{r}_j)$, proceed to the next step.
  - If Sim's running time equals 2^λ, Abort.
  - Else, increase the counter value by 1 and if counter value less than $\frac{\lambda^2}{\epsilon'}$, rewind back to the beginning of round 2 in this step (step 4).

Step 5 - Extracting Randomness:
Sim does the following:

1. For every malicious party P_j, set $\mathsf{r}_j^* = \mathsf{r}_j$ where r_j is the value sent by P_j in round 3.
2. Also, run the extractor E of the simulation extractable promise ZK, to extract the witness s_j^* used by the adversary P_j in the arguments given in round 3 and check that $\mathsf{c}_j = \mathsf{Com}(\mathsf{r}_j^*; \mathsf{s}_j^*)$. Else, output "Special Abort 1".
3. Store the pair of values $(\mathsf{r}_j^*, \mathsf{s}_j^*)$ for each malicious party P_j.
4. compute $\mathsf{r}_{\mathcal{A}} = \bigoplus_j \mathsf{r}_j^*$.

Step 6 - Forcing Output:
Sim sets $k = 1$ and does the following: (k denotes which output is being used).

1. If $k > K$, output Abort. (That is, if the number of rewinds exceeds all the answers from the ideal functionality).
2. Rewind $\mathcal{A}$ back to the beginning of round 3 in the protocol.
3. Let the number of honest parties be l. Without loss of generality, let them be $\mathsf{P}_1, \ldots, \mathsf{P}_l$.
4. Secret share ans_k into l parts $(\mathsf{r}_1, \ldots, \mathsf{r}_l)$ such that $\bigoplus_{i \in [l]} \mathsf{r}_i = (\mathsf{r}_{\mathcal{A}} \oplus \mathsf{ans}_k)$.
5. For every honest party P_i, broadcast r_i.
6. Then, for all $i \in [l]$ and for each $j \in [n]$ with $j \neq i$, do:
   - Let τ_2 denote the protocol transcript so far.
   - Continue simulating the ZK arguments as in Step 4. That is, generate $(\mathsf{prove}_3^{i \to j}) \leftarrow \mathsf{Sim}_{\mathsf{SE}}(\tau)$ for the statement $\mathsf{st}^{i \to j} = (\mathsf{c}_i, \mathsf{r}_i) \in L$.
   - Generate $(\mathsf{ver}_3^{i \to j}) \leftarrow \mathsf{Sim}_{\mathsf{SE}}(\tau_2)$.
   - Broadcast $(\{\mathsf{prove}_3^{i \to j}, \mathsf{ver}_3^{i \to j}\}_{j \neq i})$.
   - Then, after receiving $\mathcal{A}$'s messages in round 3, let $\tau_3^{j \to i}$ denote the transcript of protocol $\pi^{\mathsf{SE-PZK}}$ after round 3 with prover P_j and verifier P_i.
   - If $\mathsf{Verify}_4(\tau_3^{j \to i}, \mathsf{st}^{j \to i}) \neq 1$ where $\mathsf{st}^{j \to i} = (\mathsf{c}_j, \mathsf{r}_j)$, set $k = k + 1$ and rewind $\mathcal{A}$ back to the beginning of Step 5.
7. If all the arguments from the malicious parties verified correctly above, for each $i \in [l]$ and for each $j \in [n]$ with $j \neq i$, run the extractor E of the simulation extractable promise ZK, to extract the witness s_j used by the adversary P_j in the above argument. Let r_j be the value output in round 3. Check that $\mathsf{c}_j = \mathsf{Com}(\mathsf{r}_j; \mathsf{s}_j)$. Else, output "Special Abort 2".

8. If $(\mathsf{r}_j, \mathsf{s}_j) \neq (\mathsf{r}_j^*, \mathsf{s}_j^*)$ for any malicious party P_j, output "Special Abort 3".

9. Instruct the ideal functionality to output ans_k to the honest parties.

We now prove that the simulator is an expected PPT machine.

Claim 47. *Simulator* Sim *runs in expected time that is polynomial in* λ.

Proof. Let's analyze the running time of each step of the simulation strategy. Clearly, step 1 takes only $\mathsf{poly}(\lambda)$ time for some polynomial.

Let ϵ be the probability with which the adversary doesn't abort. That is, Sim proceeds to step 2 only with probability ϵ. Now, since the probability of the adversary not aborting is ϵ, the expected number of threads to be run by the simulator to get one non-aborting transcript is $\frac{1}{\epsilon}$. Therefore, the expected total number of threads created in step 2 is $\frac{12 \cdot \lambda}{\epsilon}$ and each thread takes only $\mathsf{poly}(\lambda)$ time.

Step 3 is trivially polynomial time.

As shown in [GK96, Lin17], the probability that the estimate ϵ' computed in step 2 is not within a factor of 2 of ϵ is at most 2^{λ}. An exact computation of how to achieve this exact bound using Chernoff bounds can be found in [HL10], Section 6.5.3. (which also explains why we chose to run step 2 till we get $12 \cdot \lambda$ non-aborting transcripts). Therefore, the number of threads created in step 4 is at most $\frac{\lambda^2}{\epsilon}$ (ignoring the constant factor). Note that step 4 might still take time 2^{λ} but this happens only when the estimate of ϵ' is incorrect: that is, when ϵ' is not within a constant factor of ϵ and this happens only with probability $2^{-\lambda}$. Further, in step 4, the simulator of the argument system $\pi^{\mathsf{SE-PZK}}$ is called with the polynomial $\frac{\lambda}{\epsilon'}$. Recall that the simulator $\mathsf{Sim}_{\mathsf{SE}}$ creates only 1 set of look-ahead threads and the number of look-ahead threads created is $\lambda \cdot \frac{1}{p}$ where $p = \frac{\epsilon'}{\lambda}$ here.

Step 5 is trivially polynomial time.

Finally, in step 6, Sim tries to force the output at most $\frac{\lambda^2}{\epsilon'}$ times and each attempt at forcing is clearly in polynomial time.

Therefore, we can bound the overall running time by :

$$\mathsf{T}_{\mathsf{Sim}} = \mathsf{poly}(\lambda) + \mathsf{poly}(\lambda) \cdot \epsilon \left(\frac{12 \cdot \lambda}{\epsilon} + \left(1 - \frac{1}{2^{\lambda}}\right) \cdot \left[\frac{\lambda^2}{\epsilon} + \frac{\lambda^2}{\epsilon} + \frac{\lambda^2}{\epsilon} \right] + \left(\frac{1}{2^{\lambda}} \right) \cdot 2^{\lambda} \right)$$

$$\leq \mathsf{poly}(\lambda)$$

for some polynomial and this concludes the analysis. □

8.2.2 Hybrids

We now show that the above simulation strategy is successful. We will show this via a series of computationally indistinguishable hybrids where the first hybrid Hyb_0 corresponds to the real world and the last hybrid Hyb_5 corresponds to the ideal world.

First, assume by contradiction that there exists an adversary $\mathcal{A}$ that can distinguish the real and ideal worlds with some non-negligible probability $1000 \cdot \mu$. We will use this value μ in the hybrids. We will show that each pair of consecutive hybrids is indistinguishable except with probability at most $100 \cdot \mu$ and this completes the proof.

- Hyb_0 **- Real World:** In this hybrid, consider a simulator $\mathsf{Sim}_{\mathsf{Hyb}}$ that plays the role of the honest parties.

- Hyb_1 **- Extracting Randomness:** In this hybrid, $\mathsf{Sim}_{\mathsf{Hyb}}$ first runs the "Check Abort" step using the honest parties' strategy - that is, step 1 in the description of Sim to check if the adversary aborts.

  Suppose the adversary doesn't cause an abort - that is, the "Check Abort" step succeeded. Then, $\mathsf{Sim}_{\mathsf{Hyb}}$ rewinds back to the end of round 1 of the protocol and performs exactly as in Hyb_0 using the honest parties' strategy. If $\mathsf{Sim}_{\mathsf{Hyb}}$ now receives an Abort at the end of round 3, it rewinds back to the end of round 1 and runs the main thread again using the honest parties' strategy. This process happens $\frac{1}{\mu}$ times. $\mathsf{Sim}_{\mathsf{Hyb}}$ then runs step 5 in the description of Sim to store the adversary's randomness.

  Observe that $\mathsf{Sim}_{\mathsf{Hyb}}$ runs in polynomial time because, by assumption, μ was non-negligible. The same argument holds for the subsequent hybrids as well.

- Hyb_2 **- Simulate ZK:** In this hybrid, in each of the $\frac{1}{\mu}$ rewound executions of the main thread, $\mathsf{Sim}_{\mathsf{Hyb}}$ computes a simulated ZK argument. Further, the simulator $\mathsf{Sim}_{\mathsf{SE}}$ of the argument system $\pi^{\mathsf{SE-PZK}}$ is invoked with the value μ as the probability of not abort.

- Hyb_3 **- Switching Commitment:** In this hybrid, in round 2 of each of the $\frac{1}{\mu}$ rewound executions of the main thread, $\mathsf{Sim}_{\mathsf{Hyb}}$ computes $\mathsf{c}_i = \mathsf{Com}(0; \mathsf{s}_i)$ for each honest party P_i.

- Hyb_4 **- Abort Probability Estimation:** In this hybrid, $\mathsf{Sim}_{\mathsf{Hyb}}$ now does exactly as done by Sim in steps 1-5. In particular, $\mathsf{Sim}_{\mathsf{Hyb}}$ no longer runs the rewound executions of the main thread $\frac{1}{\mu}$ times and instead, it is executed $\frac{\lambda^2}{\epsilon'}$ times as in the description of Sim where ϵ' is estimated in step 2. $\mathsf{Sim}_{\mathsf{Hyb}}$ also queries the ideal functionality in this hybrid to receive the list of outputs $(\mathsf{ans}_1, \ldots, \mathsf{ans}_K)$.

- Hyb_5 **- Output Forcing:** $\mathsf{Sim}_{\mathsf{Hyb}}$ now also runs step 6 of the ideal world simulator Sim to force an output. This hybrid corresponds to the ideal world.

8.2.3 Indistinguishabilty of hybrids

Claim 48. *Assuming the Simulation Extractability property of the argument system $\pi^{\mathsf{SE-PZK}}$, Hyb_0 is computationally indistinguishable from Hyb_1 except with probability at most μ.*

Proof. In Hyb_0, $\mathsf{Sim}_{\mathsf{Hyb}}$ runs the protocol once using the honest strategy. In Hyb_1, if the "Check Abort" step succeeded, $\mathsf{Sim}_{\mathsf{Hyb}}$ runs the protocol using the honest strategy $\frac{1}{\mu}$ times where μ is the adversary's distinguishing advantage in the overall experiment.

First, suppose the adversary aborts with probability greater than μ. Then, both hybrids output identical threads except with probability at most μ.

Suppose the adversary doesn't abort with probability greater than
mu, then, by Markov's inequality, except with probability μ, the adversary's view in Hyb_1 consists of a thread of execution identically distributed to Hyb_0.

Therefore, now the only remaining difference between the two hybrids is that in Hyb_1, $\mathsf{Sim}_{\mathsf{Hyb}}$ may output "Special Abort 1" which doesn't happen in Hyb_0. We will now show that $\mathsf{Pr}[\ \mathsf{Sim}_{\mathsf{Hyb}}$ outputs "Special Abort 1" $] \leq \mu$ in Hyb_1 and this completes the proof of indistinguishability between the two hybrids. Recall that in this case, $\epsilon \geq \mu$. Therefore now, in Hyb_2, since each look-ahead thread is identical to the execution in the "Abort Check" step, in each look-ahead thread, the probability with which the transcript is non-aborting is same as ϵ. Therefore, each look-ahead thread, with noticeable probability, produces non-aborting transcripts. Recall from the construct of the extractor E that E internally runs the extractor $\mathsf{Ext}_{\mathsf{NMCom}}$ and is successful if $\mathsf{Ext}_{\mathsf{NMCom}}$

is successful. From the properties of the non-malleable commitment scheme NMCom used in the construction of $\pi^{\mathsf{SE-PZK}}$, recall that $\mathsf{Ext}_{\mathsf{NMCom}}$ successfully extracts with noticeable probability given the transcript of 2 non-aborting executions. Since each look-ahead thread produces a non-aborting transcript with noticeable probability at least μ and we run $\frac{1}{\mu}$ look-ahead threads, clearly, $\mathsf{Ext}_{\mathsf{NMCom}}$ successfully extracts with noticeable probability. Hence, the extractor E of the simulation extractable promise ZK argument system successfully extracts with noticeable probability and thus, in this case too, $\mathsf{Sim}_{\mathsf{Hyb}}$ outputs "Special Abort 1" only with probability at most μ. This completes the proof. □

Claim 49. *Assuming the Simulation Security property of the Simulation Extractable Promise ZK argument* $\pi^{\mathsf{SE-PZK}}$, Hyb_1 *is computationally indistinguishable from* Hyb_2 *except with probability at most* $100 \cdot \mu$.

Proof. The only difference between the two hybrids is that in each thread, in Hyb_1, the arguments of the protocol $\pi^{\mathsf{SE-PZK}}$ are computed honestly while in Hyb_2, they are computed using the simulator $\mathsf{Sim}_{\mathsf{SE}}$ for the protocol $\pi^{\mathsf{SE-PZK}}$.

Suppose there exists an adversary that can distinguish between these two hybrids with non-negligible probability greater than $100 \cdot \mu$.

Recall that we denote by ϵ the probability with which the adversary causes the "Abort Check" step to succeed. Suppose $\epsilon \leq 100 \cdot \mu$. Then, with probability $\geq (1 - 100 \cdot \mu)$, the adversary causes the "Abort Check" step to fail in both the hybrids which contradicts our assumption that there exists an adversary $\mathcal{A}$ that can distinguish the two hybrids with some non-negligible probability $100 \cdot \mu$.

Therefore, it must be the case that $\epsilon \geq 100 \cdot \mu$. In other words, the probability with which the adversary doesn't abort - ϵ is clearly greater than the value μ with which the simulator $\mathsf{Sim}_{\mathsf{SE}}$ is invoked in Hyb_2. Therefore, from the simulation security property of the protocol $\pi^{\mathsf{SE-PZK}}$, except with probability at most $100 \cdot \mu$, the two hybrids are indistinguishable.

This contradicts our assumption and hence completes the proof.

□

Claim 50. *Assuming the hiding of the commitment scheme* Com, Hyb_2 *is computationally indistinguishable from* Hyb_3 *except with probability at most* μ.

Proof. The only difference between Hyb_2 and Hyb_3 is that in Hyb_3, in each thread, for every honest party P_i, the simulator now computes the commitment using input 0 while in Hyb_2, it was computed using the randomness r_i. Suppose there exists an adversary $\mathcal{A}$ that can distinguish between the two hybrids with non-negligible probability μ. We will use $\mathcal{A}$ to design an adversary $\mathcal{A}_{\mathsf{Hid}}$ that breaks the hiding of the commitment scheme Com.

$\mathcal{A}_{\mathsf{Hid}}$ interacts with a challenger $\mathcal{C}_{\mathsf{Hid}}$. $\mathcal{A}_{\mathsf{Hid}}$ performs the role of $\mathsf{Sim}_{\mathsf{Hyb}}$ in its interaction with $\mathcal{A}$ almost exactly as done in Hyb_2. Then, for each thread and every honest party P_i, $\mathcal{A}_{\mathsf{Hid}}$ picks a random string r_i and sends $(\mathsf{r}_i, 0)$ to a challenger $\mathcal{C}_{\mathsf{Hid}}$. $\mathcal{A}_{\mathsf{Hid}}$ receives a value c from $\mathcal{C}_{\mathsf{Hid}}$ which is either a commitment of r_i or a commitment of 0. $\mathcal{A}_{\mathsf{Hid}}$ sets this value c to be the value c_i in round 2 in its interaction with the adversary $\mathcal{A}$. The rest of the interaction with $\mathcal{A}$ is performed exactly as in Hyb_2.

Observe that the first case corresponds to Hyb_2 while the second case corresponds to Hyb_3. Therefore, if the adversary $\mathcal{A}$ can distinguish between these two hybrids with non-negligible probability p, $\mathcal{A}_{\mathsf{Hid}}$ will use the same guess to break the hiding of the commitment scheme Com with non-negligible probability μ which is a contradiction. □

Claim 51. Hyb_3 *is computationally indistinguishable from* Hyb_4 *except with probability at most* μ.

Proof. As in the proof of the 4 round MPC protocol, the only difference between the two hybrids is that in Hyb_3, $\mathsf{Sim}_{\mathsf{Hyb}}$ rewinds the main thread $\frac{1}{\mu}$ times while in Hyb_4, $\mathsf{Sim}_{\mathsf{Hyb}}$ first estimates the probability of not aborting to be ϵ' and then rewinds the main thread $\mathsf{min}(2^\lambda, \frac{\lambda^2}{\epsilon'})$ times. The rest of the proof follows in a very similar manner to the proof of claim 5.8 in [Lin17]. That is, we show that if the "Check Abort" step succeeds, the simulator fails in Hyb_4 only with negligible probability using the claim in [Lin17]. We already know that in Hyb_{10}, if the "Check Abort" step succeeds, the simulation successfully completes the execution except with negligible probability and hence, this completes the proof. □

Claim 52. *Assuming the Simulation Extractability property of the argument system $\pi^{\mathsf{SE-PZK}}$ and the binding of the commitment scheme Com, Hyb_4 is computationally indistinguishable from Hyb_5 except with probability at most μ.*

Proof. First, let's show that suppose $\mathsf{Sim}_{\mathsf{Hyb}}$ doesn't output "Special Abort 2" or "Special Abort 3" in Hyb_5, then the two hybrids are identical.

Recall that we denote by ϵ the probability with which the adversary causes the "Abort Check" step to succeed. Suppose $\epsilon \leq \mu$. Then, with probability $\geq (1-\mu)$, the adversary causes the "Abort Check" step to fail in both hybrids and they are identical. Now consider the case where $\epsilon \geq \mu$. Then, in Hyb_4, where $\mathsf{Sim}_{\mathsf{Hyb}}$ runs the rewound execution of the main thread K times (recall $K = \frac{\lambda^2}{\epsilon'}$) using random values r_i for each honest party till in one transcript, the adversary doesn't abort in at least one of them with probability close to 1. Similarly, in Hyb_5, in one of the rewound executions of step 6, the adversary doesn't abort. In these rewound executions, each honest party's value r_i still appears uniformly random and hence the two hybrids are identical.

Therefore, we now have to show that in Hyb_5:

- $\mathsf{Pr}[$ $\mathsf{Sim}_{\mathsf{Hyb}}$ outputs "Special Abort 2" $] \leq \mu$
- $\mathsf{Pr}[$ $\mathsf{Sim}_{\mathsf{Hyb}}$ outputs "Special Abort 3" $] \leq \mu$

The proof of the first claim - that is, $\mathsf{Pr}[$ $\mathsf{Sim}_{\mathsf{Hyb}}$ outputs "Special Abort 2" $] \leq \mu$ follows from the Simulation Extractability property of the argument system $\pi^{\mathsf{SE-PZK}}$ exactly as in the proof of Claim 48.

We will now show that $\mathsf{Pr}[$ $\mathsf{Sim}_{\mathsf{Hyb}}$ outputs "Special Abort 3" $] \leq \mu$. Observe that "Special Abort 3" occurs only if for some malicious party P_j, $(\mathsf{r}_j, \mathsf{s}_j) \neq (\mathsf{r}_j^*, \mathsf{s}_j^*)$. However, since neither "Special Abort 1" nor "Special Abort 2" has occurred at this point, it must be the case that $\mathsf{c}_j = \mathsf{Com}(\mathsf{r}_j^*, \mathsf{s}_j^*)$ and $\mathsf{c}_j = \mathsf{Com}(\mathsf{r}_j, \mathsf{s}_j)$. However, this would then break the binding property of the commitment scheme Com. Formally, if there exists an adversary $\mathcal{A}$ that causes $\mathsf{Sim}_{\mathsf{Hyb}}$ to output "Special Abort 3" in Hyb_5 with non-negligible probability μ, then we can construct a reduction as follows. The reduction interacts with $\mathcal{A}$ exactly as done by $\mathsf{Sim}_{\mathsf{Hyb}}$ in Hyb_5 and when $\mathsf{Sim}_{\mathsf{Hyb}}$ outputs "Special Abort 3", the reduction outputs the two pairs of values $(\mathsf{r}_j, \mathsf{s}_j)$ and $(\mathsf{r}_j^*, \mathsf{s}_j^*)$ to the challenger of the commitment scheme thus breaking the binding property with probability at least μ. This completes the proof. □

9 Three Round Secure Computation of Input-less Randomized Functionalities

In this section, we generalize the results from the previous section to the setting of any input-less randomized functionality. That is, similar to coin tossing, we first define the notion of "List

Computation" for input-less randomized functionalities and then show how to securely achieve it in three rounds.

Definition 11. *An n party protocol π^f for any input-less randomized functionality f in the simultaneous message setting is said to "List Compute" f securely if for every PPT adversary $\mathcal{A}$ corrupting at most $(n1)$ parties, there exists an expected PPT simulator $\mathcal{S}$ and a polynomial p such that the output of the experiments* REAL *and* IDEAL *defined below are indistinguishable. In the real world, we denote the result of running protocol π with adversary $\mathcal{A}$ as a pair $(\mathsf{c}, \mathsf{view}_{\mathcal{A}})$ where $\mathsf{c} \in \{0,1\}^l \cup \{\bot\}$ is the output of the protocol and $\mathsf{view}_{\mathcal{A}}$ is the view of the adversary $\mathcal{A}$. Similarly, we use the pair $(\tilde{\mathsf{c}}, \mathsf{view}_{\mathcal{S}})$ in the ideal world.*

We use l to denote the output length of the protocol.

$\mathsf{REAL}(1^\lambda, 1^l)$	$\mathsf{IDEAL}(1^\lambda, 1^l)$
$(\mathsf{c}, \mathsf{view}_{\mathcal{A}}) \leftarrow \mathsf{REAL}_{\pi^f, \mathcal{A}}(1^\lambda, 1^l)$	Pick $(\mathsf{c}_1, \ldots, \mathsf{c}_{p(\lambda)})$ randomly from the output distribution of f.
	$(\tilde{\mathsf{c}}, \mathsf{view}_{\mathcal{S}}) \leftarrow \mathcal{S}^{\mathcal{A}}(\mathsf{c}_1, \ldots, \mathsf{c}_{p(\lambda)}, 1^\lambda, 1^l)$
Output $(\mathsf{c}, \mathsf{view}_{\mathcal{A}})$	If $\tilde{\mathsf{c}} \in \{\mathsf{c}_1, \ldots, \mathsf{c}_{p(\lambda)}\}$, then output $(\tilde{\mathsf{c}}, \mathsf{view}_{\mathcal{S}})$
	Else, output fail.

Table 2: List Computation of any Input-less Randomized Functionalitt f

9.1 Construction

The construction is very similar to the one for List Coin Tossing with the only difference being that instead of each party committing to its randomness r_i in the second round and opening it in the third round, we now run a 3 round semi-malicious secure MPC protocol (that is secure against malicious adversaries in the first round) for the input-less randomized functionality f. Recall that such a protocol exists assuming polynomially secure LWE [BHP17]. We describe the protocol in Figure 11 for the case of completeness.

Consider n parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ who wish to "List Compute" an input-less randomized functionality f. In this section, we prove the following theorem:

Theorem 12. *Assuming the existence of polynomially secure LWE, the protocol π^{RF} presented in Figure 11 is a 3 round protocol that "List Computes" any input-less randomized functionality f in the plain model secure against any adversary that corrupts up to $(n-1)$ parties.*

NP Languages. In our construction, we use proofs for NP language L characterized by the following relation R.
Statement : $\mathsf{st} = (\mathsf{msg}_2, \mathsf{Trans}_1, \mathsf{msg}_3, \mathsf{Trans}_2)$
Witness : $\mathsf{w} = (\mathsf{s})$
$R(\mathsf{st}, \mathsf{w}) = 1$ if and only if: The messages $(\mathsf{msg}_2, \mathsf{msg}_3)$ are generated by running protocol π^{SM} using randomness s where the protocol transcript at the end of round 1 is Trans_1 and after round 2 is Trans_2.

Formally, $R(\mathsf{st}, \mathsf{w}) = 1$ if and only if :

- $\mathsf{msg}_2 = \pi_2^{\mathsf{SM}}(\bot, \mathsf{Trans}_1; \mathsf{s})$ AND
- $\mathsf{msg}_3 = \pi_3^{\mathsf{SM}}(\bot, \mathsf{Trans}_2; \mathsf{s})$

1. **Round 1:** P_i does the following:
   - For each $j \in [n]$ with $j \neq i$, generate $(\mathsf{prove}_1^{i\to j}) \leftarrow \mathsf{Prove}_1(1^\lambda)$ and $(\mathsf{ver}_1^{i\to j}) \leftarrow \mathsf{Verify}_1(1^\lambda)$.
   - Compute $\mathsf{msg}_{1,i} \leftarrow \pi_1^{\mathsf{SM}}(\bot; \mathsf{s}_i)$ where s_i is picked randomly.
   - Broadcast $(\mathsf{msg}_{1,i}, \{\mathsf{prove}_1^{i\to j}, \mathsf{ver}_1^{i\to j}\}_{j\neq i})$.
2. **Round 2:** P_i does the following:
   - For each $j \in [n]$ with $j \neq i$, do:
     - Let $\tau_1^{i\to j}$ denote the transcript of protocol $\pi^{\mathsf{SE-PZK}}$ after round 1 with prover P_i and verifier P_j. That is, $\tau_1^{i\to j} = (\mathsf{prove}_1^{i\to j}, \mathsf{ver}_1^{j\to i})$.
     - Similarly, let $\tau_1^{j\to i}$ denote the transcript of protocol $\pi^{\mathsf{SE-PZK}}$ after round 1 with prover P_j and verifier P_i.
     - Generate $(\mathsf{prove}_2^{i\to j}) \leftarrow \mathsf{Prove}_2(\tau_1^{i\to j})$ and $(\mathsf{ver}_2^{i\to j}) \leftarrow \mathsf{Verify}_2(\tau_1^{j\to i})$.
   - Let Trans_1 denote the transcript of protocol π^{SM} after round 1. Compute $\mathsf{msg}_{2,i} \leftarrow \pi_2^{\mathsf{SM}}(\bot, \mathsf{Trans}_1; \mathsf{s}_i)$.
   - Broadcast $(\mathsf{msg}_{2,i}, \{\mathsf{prove}_2^{i\to j}, \mathsf{ver}_2^{i\to j}\}_{j\neq i})$.
3. **Round 3:** P_i does the following:
   - Let Trans_2 denote the transcript of protocol π^{SM} after round 2. Compute $\mathsf{msg}_{3,i} \leftarrow \pi_2^{\mathsf{SM}}(\bot, \mathsf{Trans}_2; \mathsf{s}_i)$.
   - For each $j \in [n]$ with $j \neq i$, do:
     - Let $\tau_2^{i\to j}$ denote the transcript of protocol $\pi^{\mathsf{SE-PZK}}$ after round 2 with prover P_i and verifier P_j.
     - Similarly, let $\tau_2^{j\to i}$ denote the transcript of protocol $\pi^{\mathsf{SE-PZK}}$ after round 2 with prover P_j and verifier P_i.
     - Generate $(\mathsf{prove}_3^{i\to j}) \leftarrow \mathsf{Prove}_3(\tau_2^{i\to j}, \mathsf{st}^{i\to j}, \mathsf{w}^{i\to j})$ for the statement $\mathsf{st}^{i\to j} = (\mathsf{msg}_{2,i}, \mathsf{Trans}_1, \mathsf{msg}_{3,i}, \mathsf{Trans}_2) \in L$ using witness $\mathsf{w}^{i\to j} = \mathsf{s}_i$.
     - Generate $(\mathsf{ver}_3^{i\to j}) \leftarrow \mathsf{Verify}_3(\tau_2^{j\to i})$.
   - Broadcast $(\mathsf{msg}_{3,i}, \{\mathsf{prove}_3^{i\to j}, \mathsf{ver}_3^{i\to j}\}_{j\neq i})$.
4. **Output Computation:** P_i does the following:
   - For each $j \in [n]$ with $j \neq i$, do:
     - Let $\tau_3^{j\to i}$ denote the transcript of protocol $\pi^{\mathsf{SE-PZK}}$ after round 3 with prover P_j and verifier P_i.
     - Abort if $\mathsf{Verify}_4(\tau_3^{j\to i}, \mathsf{st}^{j\to i}) \neq 1$ where $\mathsf{st}^{j\to i} = (\mathsf{msg}_{2,j}, \mathsf{Trans}_1, \mathsf{msg}_{3,j}, \mathsf{Trans}_2)$. In particular, send a global abort signal to all parties so that everyone aborts.
   - Else, compute output $\mathsf{y}_i \leftarrow \mathsf{OUT}(\bot, \mathsf{Trans}_3; \mathsf{s}_i)$ where Trans_3 is the transcript of protocol π^{SM} after round 3.

Figure 11: 3 round malicious secure protocol π^{RF}.

9.2 Security Proof

The proof of security is very similar to the "List Coin Tossing" security proof in Section 8.2. The only difference is that instead of relying on the hiding of the non-interactive commitment scheme (which was the case in the List Coin Tossing proof), we will now rely on the security of the 3 round semi-malicious secure protocol π^{SM}. That is, in the proof from Section 8.2, in Hyb_3, instead of switching the commitment to using input 0, we will switch the messages of protocol π^{SM} to the simulated messages. Similarly, the outputs forced in the final hybrid will now be from the output distribution of f (obtained from the ideal functionality).

References

[ACJ17] Prabhanjan Ananth, Arka Rai Choudhuri, and Abhishek Jain. A new approach to round-optimal secure multiparty computation. In *CRYPTO*, pages 468–499, 2017. 4, 6, 7, 9, 10, 11, 25

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *EUROCRYPT*, pages 483–501, 2012. 7, 90

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 106–115. IEEE, 2001. 3

[BGI+17] Saikrishna Badrinarayanan, Sanjam Garg, Yuval Ishai, Amit Sahai, and Akshay Wadia. Two-message witness indistinguishability and secure computation in the plain model from new assumptions. In *ASIACRYPT*, 2017. 7

[BGJ+17] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Dakshita Khurana, and Amit Sahai. Round optimal concurrent mpc via strong simulation. In *TCC*, 2017. 7

[BHP17] Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. In *TCC*, 2017. 4, 5, 6, 7, 33, 82

[BKP17] Nir Bitansky, Yael Tauman Kalai, and Omer Paneth. Multi-collision resistance: A paradigm for keyless hash functions. *IACR Cryptology ePrint Archive*, 2017:488, 2017. 3

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *STOC*, pages 503–513, 1990. 4, 6

[BS05] Boaz Barak and Amit Sahai. How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*, pages 543–552, 2005. 9

[Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In Juris Hartmanis, editor, *STOC*, pages 364–369. ACM, 1986. 4

[COSV16] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Concurrent non-malleable commitments (and more) in 3 rounds. In *CRYPTO*, pages 270–299, 2016. 6

[COSV17a] Michele Ciampi, Rafail Ostrovsky, Siniscalchi, and Ivan Visconti. Delayed-input non-malleable zero knowledge and multi-party coin tossing in four rounds. In *TCC*, 2017. 4, 7, 14

[COSV17b] Michele Ciampi, Rafail Ostrovsky, Siniscalchi, and Ivan Visconti. Round-optimal secure two-party computation from trapdoor permutations. In *TCC*, 2017. 4, 7

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *STOC*, pages 542–552, 1991. 9, 13

[DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *FOCS*, pages 283–293, 2000. 7

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *FOCS*, pages 308–317, 1990. 8

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In *TCC*, pages 74–94, 2014. 7

[GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *J. Cryptology*, 9(3):167–190, 1996. 3, 4, 9, 12, 43, 74, 78

[GKP17] Sanjam Garg, Susumu Kiyoshima, and Omkant Pandey. On the exact round complexity of self-composable two-party computation. In *EUROCRYPT*, pages 194–224, 2017. 7

[GKS16] Vipul Goyal, Dakshita Khurana, and Amit Sahai. Breaking the three round barrier for non-malleable commitments. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 21–30. IEEE Computer Society, 2016. 14

[GMPP16] Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. In *EUROCRYPT*, pages 448–476, 2016. 3, 4, 5, 6, 7, 9, 12, 73

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 1989. 3

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987. 4, 9

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004. 14, 88

[Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In *STOC*, pages 695–704, 2011. 4, 7, 13

[GPR16] Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. In *STOC*, pages 1128–1141, 2016. 9, 14, 34, 60

[GRRV14] Vipul Goyal, Silas Richelson, Alon Rosen, and Margarita Vald. An algebraic approach to non-malleability. In *FOCS*, pages 41–50, 2014. 16

[GS17] Sanjam Garg and Akshayaram Srinivasan. Garbled protocols and two-round mpc from bilinear maps. In *FOCS*, 2017. 7

[HHPV17] Shai Halevi, Carmit Hazay, Antigoni Polychroniadou, and Muthuramakrishnan Venkitasubramaniam. Round-optimal secure multi-party computation. *IACR Cryptology ePrint Archive*, 2017:1056, 2017. 6

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols - Techniques and Constructions*. Information Security and Cryptography. Springer, 2010. 43, 78

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *STOC*, pages 21–30, 2007. 90

[JKKR17] Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Ron Rothblum. Distinguisher-dependent simulation in two rounds and its applications. In *CRYPTO*, 2017. 4, 7, 9, 11, 17, 34, 92, 93

[Khu17] Dakshita Khurana. Round optimal concurrent non-malleability from polynomial hardness. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 139–171. Springer, 2017. 9

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *CRYPTO*, pages 335–354, 2004. 4, 5, 6, 73

[KOS03] Jonathan Katz, Rafail Ostrovsky, and Adam D. Smith. Round efficiency of multi-party computation with a dishonest majority. In *EUROCRYPT*, pages 578–595, 2003. 4, 7

[KS17] Dakshita Khurana and Amit Sahai. Two-message non-malleable commitments from standard sub-exponential assumptions. *IACR Cryptology ePrint Archive*, 2017:291, 2017. 7

[Lin17] Yehuda Lindell. How to simulate it - A tutorial on the simulation proof technique. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography.*, pages 277–346. Springer International Publishing, 2017. 43, 58, 74, 78, 81

[LPV08] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. Concurrent non-malleable commitments from any one-way function. In *TCC*, pages 571–588, 2008. 13

[LS90] Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In *CRYPTO*, pages 353–365, 1990. 90

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In *EUROCRYPT*, pages 735–763, 2016. 7

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 232–241, 2004. 4, 7

[PPV08] Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive one-way functions and applications. In *CRYPTO*, pages 57–74, 2008. 7

[PR05] Rafael Pass and Alon Rosen. Concurrent non-malleable commitments. In *FOCS*, pages 563–572, 2005. 13

[PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *FOCS*, pages 366–375, 2002. 8, 14, 25

[PW10] Rafael Pass and Hoeteck Wee. Constant-round non-malleable commitments from sub-exponential one-way functions. In *EUROCRYPT*, pages 638–655, 2010. 4, 7

[Ros04] Alon Rosen. A note on constant-round zero-knowledge proofs for NP. In *TCC*, pages 191–202, 2004. 8, 14, 25

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, pages 543–553, 1999. 9

[Wee10] Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *FOCS*, pages 531–540, 2010. 4, 7

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS*, 1982. 4

A Secure Multiparty Computation

Parts of this section have been taken verbatim from [Gol04].

A multi-party protocol is cast by specifying a random process that maps pairs of inputs to pairs of outputs (one for each party). We refer to such a process as a functionality. The security of a protocol is defined with respect to a functionality f. In particular, let n denote the number of parties. A non-reactive n-party functionality f is a (possibly randomized) mapping of n inputs to n outputs. A multiparty protocol with security parameter λ for computing a non-reactive functionality f is a protocol running in time $\mathsf{poly}(\lambda)$ and satisfying the following correctness requirement: if parties $P_1, \ldots, P_n$ with inputs $(x_1, \ldots, x_n)$ respectively, all run an honest execution of the protocol, then the joint distribution of the outputs $y_1, \ldots, y_n$ of the parties is statistically close to $f(x_1, \ldots, x_n)$.

A reactive functionality f is a sequence of non-reactive functionalities $f = (f_1, \ldots, f_\ell)$ computed in a stateful fashion in a series of phases. Let x_i^j denote the input of P_i in phase j, and let s^j denote the state of the computation after phase j. Computation of f proceeds by setting s^0 equal to the empty string and then computing $(y_1^j, \ldots, y_n^j, s^j) \leftarrow f_j(s^{j-1}, x_1^j, \ldots, x_n^j)$ for $j \in [\ell]$, where y_i^j denotes the output of P_i at the end of phase j. A multi-party protocol computing f also runs in ℓ phases, at the beginning of which each party holds an input and at the end of which each party obtains an output. (Note that parties may wait to decide on their phase-j input until the beginning of that phase.) Parties maintain state throughout the entire execution. The correctness requirement is that, in an honest execution of the protocol, the joint distribution of all the outputs $\{y_1^j, \ldots, y_n^j\}_{j=1}^{\ell}$ of all the phases is statistically close to the joint distribution of all the outputs of all the phases in a computation of f on the same inputs used by the parties.

Defining Security. We assume that readers are familiar with standard simulation-based definitions of secure multi-party computation in the standalone setting. We provide a self-contained definition for completeness and refer to [Gol04] for a more complete description. The security of a protocol (with respect to a functionality f) is defined by comparing the real-world execution of the protocol with an ideal-world evaluation of f by a trusted party. More concretely, it is required that for every adversary $\mathcal{A}$, which attacks the real execution of the protocol, there exist an adversary Sim, also referred to as a simulator, which can *achieve the same effect* in the ideal-world. Let's denote $\mathbf{x} = (x_1, \ldots, x_n)$.

The real execution In the real execution of the n-party protocol π for computing f is executed in the presence of an adversary $\mathcal{A}$. The honest parties follow the instructions of π. The adversary $\mathcal{A}$ takes as input the security parameter k, the set $I \subset [n]$ of corrupted parties, the inputs of the

corrupted parties, and an auxiliary input z. $\mathcal{A}$ sends all messages in place of corrupted parties and may follow an arbitrary polynomial-time strategy.

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

The ideal execution – security with abort . In this second variant of the ideal model, fairness and output delivery are no longer guaranteed. This is the standard relaxation used when a strict majority of honest parties is not assumed. In this case, an ideal execution for a function f proceeds as follows:

- **Send inputs to the trusted party:** As before, the parties send their inputs to the trusted party, and we let x_i' denote the value sent by P_i. Once again, for a semi-honest adversary we require $x_i' = x_i$ for all $i \in I$.
- **Trusted party sends output to the adversary:** The trusted party computes $f(x_1', \ldots, x_n') = (y_1, \ldots, y_n)$ and sends $\{y_i\}_{i\in I}$ to the adversary.
- **Adversary instructs trust party to abort or continue:** This is formalized by having the adversary send either a continue or abort message to the trusted party. (A semi-honest adversary never aborts.) In the latter case, the trusted party sends to each uncorrupted party P_i its output value y_i. In the former case, the trusted party sends the special symbol $\perp$ to each uncorrupted party.
- **Outputs:** Sim outputs an arbitrary function of its view, and the honest parties output the values obtained from the trusted party.

The interaction of Sim with the trusted party defines a random variable $\mathsf{IDEAL}_{f_\perp,\mathcal{A}(z)}(k,\mathbf{x})$ as above,and we let $\{\mathsf{IDEAL}_{f_\perp,\mathcal{A}(z),I}(k,\mathbf{x})\}_{k\in\mathsf{N},\langle\mathbf{x},z\rangle\in\{0,1\}^*}$ where the subscript "$\perp$" indicates that the adversary can abort computation of f.

Having defined the real and the ideal worlds, we now proceed to define our notion of security.

Definition 12. *Let k be the security parameter. Let f be an n-party randomized functionality, and π be an n-party protocol for $n \in \mathsf{N}$.*

1. *We say that π t-securely computes f in the presence of malicious (resp., semi-honest) adversaries if for every PPT adversary (resp., semi-honest adversary) $\mathcal{A}$ there exists a PPT adversary (resp., semi-honest adversary) Sim such that for any $I \subset [n]$ with $|I| \leq t$ the following quantity is negligible:*

$$|Pr[\mathsf{REAL}_{\pi,\mathcal{A}(z),I}(k,\mathbf{x}) = 1] - Pr[\mathsf{IDEAL}_{f,\mathcal{A}(z),I}(k,\mathbf{x}) = 1]|$$

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

2. *Similarly, π t-securely computes f* with abort *in the presence of malicious adversaries if for every PPT adversary $\mathcal{A}$ there exists a polynomial time adversary Sim such that for any $I \subset [n]$ with $|I| \leq t$ the following quantity is negligible:*

$$|Pr[\mathsf{REAL}_{\pi,\mathcal{A}(z),I}(k,\mathbf{x}) = 1] - Pr[\mathsf{IDEAL}_{f_\perp,\mathcal{A}(z),I}(k,\mathbf{x}) = 1]|.$$

Security Against Semi-Malicious Adversaries We take this definition almost verbatim from [AJL+12]. We define a notion of a semi-malicious adversary that is stronger than the standard notion of semi-honest adversary and formalize security against semi-malicious adversaries. A semi-malicious adversary is modeled as an interactive Turing machine (ITM) which, in addition to the standard tapes, has a special witness tape. In each round of the protocol, whenever the adversary produces a new protocol message msg on behalf of some party P_k, it must also write to its special witness tape some pair (x, r) of input x and randomness r that explains its behavior. More specifically, all of the protocol messages sent by the adversary on behalf of P_k up to that point, including the new message m, must exactly match the honest protocol specification for P_k when executed with input x and randomness r. Note that the witnesses given in different rounds need not be consistent. Also, we assume that the attacker is rushing and hence may choose the message m and the witness (x, r) in each round adaptively, after seeing the protocol messages of the honest parties in that round (and all prior rounds). Lastly, the adversary may also choose to abort the execution on behalf of P_k in any step of the interaction.

Definition 13. *We say that a protocol π securely realizes f for semi-malicious adversaries if it satisfies Definition 12 when we only quantify over all semi-malicious adversaries A.*

B WI with Bounded Rewinding Security

In this section, for any constant $L > 0$, assuming the existence of injective one way functions, we construct a 3 round delayed-input witness indistinguishable argument with L-rewinding security as defined in Section 4.

High Level Overview. At a very high level, the protocol consists of combining the 3 round delayed input WI protocol in [LS90] with the bounded rewinding secure 3 round "MPC in the head" based 3 round ZK protocol of [IKOS07]. The main idea being that in the WI protocol, when the challenge is 0, we will respond by invoking the ZK protocol that has rewinding security and when the challenge is 1, we respond as in the WI protocol itself. Now note that to recover the entire witness, you need to learn the full outputs of both challenges. However, for the challenge 0 case, even with a bounded number of rewinds, from the security of the ZK protocol, the witness is computationally hidden.

Construction. The protocol RWI consists of 4 algorithm $(\mathsf{RWI}_1, \mathsf{RWI}_2, \mathsf{RWI}_3, \mathsf{RWI}_4)$ where the first 3 denote the algorithms used by the prover and verifier to send their messages and the last one is the final verification algorithm. We use two building blocks. The first being a 3 round delayed input witness indistinguishable argument $\mathsf{WI} = (\mathsf{WI}_1, \mathsf{WI}_2, \mathsf{WI}_3, \mathsf{WI}_4)$ from the work of Lapidot and Shamir [LS90] that is based on injective one way functions. We will open up this construction for ease of exposition. The second being a 3 round zero knowledge protocol with constant soundness error using the "MPC in the Head" approach in the paper of Ishai et al. [IKOS07] which is once again based on injective one way functions. We refer to this protocol as $\mathsf{Head.ZK} = (\mathsf{Head.ZK}_1, \mathsf{Head.ZK}_2, \mathsf{Head.ZK}_3, \mathsf{Head.ZK}_4)$. Recall from [IKOS07] that the prover runs a t-private MPC protocol amongst n parties and the verifier asks for 2 views. That is, the protocol is sound even if the verifier opens only 2 views and even given k-views, a malicious verifier can't distinguish honestly generated views of the other honest parties from simulated ones. We will invoke the ZK protocol with $t \geq 2 \cdot L$.

We will construct the protocol RWI with constant soundness error. for the Graph Hamiltonicity problem. This is described formally in Figure 12. Ofcourse, for any NP Language L, we can get

a proof by applying a Karp reduction to this problem. Also, we know that WI arguments can be composed to decrease the soundness error probability to negligible and the same works for our setting too.

Inputs: Prover P has input an efficiently sampleable distribution of graphs G (the statement) and their corresponding hamiltonian cycles (which form the witness).

1. **Round 1: Prover message:**
   - Pick a random cyclic graph H and compute $\mathsf{c} = \mathsf{Com}(H;\mathsf{r})$ using a non-interactive commitment. That is, this is a commitment to the adjacency matrix of H.
   - Compute $\mathsf{hzk}_1^{P\to V} \leftarrow \mathsf{Head.ZK}_1(1^\lambda)$ and send $(\mathsf{c}, \mathsf{prove}_1^{P\to V})$ to V.
2. **Round 2: Verifier message:**
   - Pick a random bit b.
   - If $b = 0$, compute $\leftarrow \mathsf{hzk}_2^{V\to P} \leftarrow \mathsf{Head.ZK}_2(\mathsf{hzk}_1^{P\to V})$ and send $(0, \mathsf{hzk}_2^{V\to P})$ to V.
   - If $b = 1$, send $(1, \perp)$ to V.
3. **Round 3: Prover message:**
   - Sample a graph G and an associated hamiltonian cycle on it from the efficiently sampleable distribution. Send G to V.
   - If $b = 0$, compute and send $\mathsf{hzk}_3^{P\to V} \leftarrow \mathsf{Head.ZK}_3(\mathsf{hzk}_1^{P\to V}, \mathsf{hzk}_1^{V\to P}, \mathsf{st} = (\mathsf{c}), w = (H,\mathsf{r}))$ where the statement $\mathsf{st} = (\mathsf{c})$ is in the language if there exists a witness $w = (H,\mathsf{r})$ such that H is a cyclic graph and $\mathsf{c} = \mathsf{Com}(H;\mathsf{r})$.
   - If $b = 1$, compute and send a permutation π that maps H onto the hamiltonian cycle of G. Also, decommit to the set of non-edges in the adjacency graph of H.
4. **Verifier Output:**
   If $b = 0$, run the algorithm $\mathsf{Head.ZK}_4$ and output whatever it outputs. If $b = 1$, check that the output corresponds to a valid set of non-edges in H and that exactly n of them are not opened where n is the number of vertices. (that is, essentially run the algorithm WI_4 for the $b = 1$ case alone.)

Figure 12: 3 round Bounded Rewinding Secure WI

Proof Sketch. We now briefly describe the proof of security.

Soundness:
Follows from the soundness of the 3 round witness indistinguishable protocol WI and the soundness of the zero knowledge protocol Head.ZK.

Bounded Rewinding Security:
Note that each query of the malicious verifier V^* can be split into two parts - the first is a bit b that is either 0 or 1 and if $b = 0$, the second part consists of 2 views. The simulator, guesses the set of queries made across all the L-rewinds in advance and then just runs the underlying simulators of the protocols WI and Head.ZK appropriately in each query depending on whether the query was for $b = 0$ or $b = 1$.

C MPC - Proof of Aborting Case

Recall that the only difference between the two hybrids is if the adversary causes $\mathsf{Sim}_{\mathsf{Hyb}}$ to abort at the end of round 3 - that is, the "Check Abort" step doesn't succeed in Hyb_1. In that case, in Hyb_0, $\mathsf{Sim}_{\mathsf{Hyb}}$ uses the honest parties' inputs to run the protocol while in Hyb_1, $\mathsf{Sim}_{\mathsf{Hyb}}$ runs the protocol using input 0 for every honest party. We will now show that these two hybrids are indistinguishable via a sequence of intermediate hybrids H_0 to H_{12}. Here, H_0 will correspond to Hyb_0 and H_{12} will correspond to Hyb_1.

- H_0 : This is same as Hyb_0.
- H_1 **- Simulate Weak ZK:** In this hybrid, $\mathsf{Sim}_{\mathsf{Hyb}}$ creates two sets of look-ahead threads that run only round 3 of the protocol. The first set is identical to the main thread in Hyb_0. The second set is identical to the main thread in Hyb_1 (same as H_{12}) - that is, it computes an extractable commitment using input 0 and the messages for the underlying semi-malicious MPC protocol π^{SM} using input 0.

  These look-ahead threads are used to simulate the Weak ZK argument WZK. Using all these look-ahead threads, $\mathsf{Sim}_{\mathsf{Hyb}}$ runs the algorithm $\mathsf{Sim}_{\mathsf{WZK}}$ of the weak ZK protocol to simulate the weak ZK argument given by every honest party in round 3 of the main thread. Note that in the look-ahead threads, the weak ZK is computed honestly.

  Recall from [JKKR17] that $\mathsf{Sim}_{\mathsf{WZK}}$ is a distinguish dependent simulator that uses the distinguisher's response on all these look-ahead threads to simulate the weak ZK. Note that these look-ahead threads that are created here are local to this proof and have nothing to do with the look-ahead threads that are created by the actual simulator Sim for the overall protocol. In particular, these local look-ahead threads are not created by Sim in the overall proof. As a result, though $\mathsf{Sim}_{\mathsf{WZK}}$ depends on the distinguisher, we use it only in the underlying reductions here and not in the overall simulation and hence our overall simulator Sim is not distinguisher dependent.
- H_2 **- Change Second ExtCom:** In the main thread, for each honest party P_i, $\mathsf{Sim}_{\mathsf{Hyb}}$ does the following: in round 3, compute $\mathsf{r.ec}_{b,3}^{i\to j} = \mathsf{R.ECom}_3(\mathsf{x}_i, \mathsf{r}_i, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}; \mathsf{r}_{b,\mathsf{r.ec}}^{i\to j})$. That is, in the main thread, the simulator now commits to the input in the second extractable commitment scheme too.

  Further, $\mathsf{Sim}_{\mathsf{Hyb}}$ also changes the look-ahead threads. In the first set of look-ahead threads, the second extractable commitment is computed as in the main thread: that is, $\mathsf{r.ec}_{b,3}^{i\to j} = \mathsf{R.ECom}_3(\mathsf{x}_i, \mathsf{r}_i, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}; \mathsf{r}_{b,\mathsf{r.ec}}^{i\to j})$. In the second set of look-aheads, the second extractable commitment is computed using input 0: that is, $\mathsf{r.ec}_{b,3}^{i\to j} = \mathsf{R.ECom}_3(0, \mathsf{r}_i, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}; \mathsf{r}_{b,\mathsf{r.ec}}^{i\to j})$ where r_i is sampled fresh in each look-ahead thread.
- H_3 **- Switch WI:** In every thread (main and look-aheads), in round 3, $\mathsf{Sim}_{\mathsf{Hyb}}$ computes the rewinding secure WI from each honest party P_i to malicious party P_j using the witness for the second statement: that is, using the second extractable commitment.
- H_4 **- Switch Commitment:** In the main thread, in round 3, $\mathsf{Sim}_{\mathsf{Hyb}}$ computes $\mathsf{nc}_i \leftarrow \mathsf{NCom}(0; \mathsf{r}_{\mathsf{nc},i})$. Note that in the look ahead threads, nc_i continues to be computed as a commitment to 1 so that the weak ZK arguments in the look-ahead threads can be honestly generated.

- H_5 - **Switch WI:** In the main thread, in round 3, $\mathsf{Sim}_{\mathsf{Hyb}}$ computes the rewinding secure WI from each honest party P_i to malicious party P_j using the witness for the fourth statement: that is, using the commitment nc_i to input 0.

- H_6 - **Change First ExtCom:** In the main thread, for each honest party P_i, $\mathsf{Sim}_{\mathsf{Hyb}}$ does the following: in round 3, compute $\mathsf{r.ec}_{a,3}^{i\to j} = \mathsf{R.ECom}_3(0, \mathsf{r}_i, \mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{a,2}^{j\to i}; \mathsf{r}_{a,\mathsf{r.ec}}^{i\to j})$. That is, in the main thread, the simulator now commits to input 0 in the first extractable commitment scheme.

- H_7 - **Change π^{SM}:** In the main thread, for every honest party P_i, $\mathsf{Sim}_{\mathsf{Hyb}}$ compute $\mathsf{msg}_{2,i} \leftarrow \mathcal{S}_2(\mathsf{Trans}_2; \mathsf{r}_i)$. Note that this is same as computing $\mathsf{msg}_{2,i} \leftarrow \pi_2^{\mathsf{SM}}(0, \mathsf{Trans}_1; \mathsf{r}_i)$.

- H_8 - **Switch WI:** In every thread (main and look-aheads), in round 3, $\mathsf{Sim}_{\mathsf{Hyb}}$ computes the rewinding secure WI from each honest party P_i to malicious party P_j using the witness for the first statement: that is, using the first extractable commitment.

- H_9 - **Change Second ExtCom:** In every thread (main and look-aheads), for each honest party P_i, $\mathsf{Sim}_{\mathsf{Hyb}}$ does the following: in round 3, compute $\mathsf{r.ec}_{b,3}^{i\to j} = \mathsf{R.ECom}_3(\bot, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}; \mathsf{r}_{b,\mathsf{r.ec}}^{i\to j})$. That is, the simulator now commits to bot in the second extractable commitment scheme.

- H_{10} - **Switch WI:** In the main thread, in round 3, $\mathsf{Sim}_{\mathsf{Hyb}}$ computes the rewinding secure WI from each honest party P_i to malicious party P_j using the witness for the first statement: that is, using the first extractable commitment.

- H_{11} - **Switch Commitment:** In the main thread, in round 3, $\mathsf{Sim}_{\mathsf{Hyb}}$ computes $\mathsf{nc}_i \leftarrow \mathsf{NCom}(1; \mathsf{r}_{\mathsf{nc},i})$.

- H_{12} - **Stop Simulating Weak ZK:** In this hybrid, $\mathsf{Sim}_{\mathsf{Hyb}}$ stops the two sets of look-ahead threads and computes the Weak ZK argument honestly. This hybrid exactly corresponds to Hyb_1.

We now show that each pair of successive hybrids in the above list is computationally indistinguishable and that completes the proof of Claim 10.

Sub-Claim 5. *Assuming the Weak Zero Knowledge property of the argument system* WZK, H_0 *is computationally indistinguishable from* H_1.

Proof. The only difference between the two hybrids is that in H_0, the Weak ZK argument is computed honestly while in H_1, the Weak ZK argument on the main thread is computed using the simulator $\mathsf{Sim}_{\mathsf{WZK}}$ from [JKKR17]. The proof of indistinguishability of the two hybrids directly follows from the security of the Weak ZK argument system constructed in [JKKR17].

Here, note that for the second set of look-ahead threads, we can generate the extractable commitment using input 0 even though the first 2 rounds are already fixed because the scheme is delayed input and allows sampling a fresh value α for each third round message. That is, in round 3 of each look-ahead thread in the second set, corresponding to the commitment from every honest party P_i to malicious party P_j, $\mathsf{Sim}_{\mathsf{Hyb}}$ computes $\mathsf{r.ec}_{b,3}^{i\to j}$ as $(\alpha^{i\to j}, \mathsf{brew.ec}_{b,3}^{i\to j}, \beta^{i\to j})$ where $\mathsf{brew.ec}_{b,3}^{i\to j} = \mathsf{BRew.ECom}_3(\mathsf{r}_{i,j}, \mathsf{brew.ec}_{b,1}^{i\to j}, \mathsf{brew.ec}_{b,2}^{j\to i}; \mathsf{r}_{b,\mathsf{brew.ec}})$, $\beta^{i\to j} = \mathsf{PRF}(\mathsf{r}_{i,j}, \alpha^{i\to j}) \oplus 0$ and $\mathsf{r}_{i,j}, \mathsf{r}_{b,\mathsf{brew.ec}}$ are the values picked uniformly at random common with the main thread. Note that only $\alpha^{i\to j}$ is sampled afresh in each thread. Also, note that the second round message of protocol π^{SM} can also be generated using input 0 in the second set of look-ahead threads because the first round message doesn't depend on the input at all. □

Sub-Claim 6. *Assuming the security of the extractable commitment* $\mathsf{BRew.ECom}$ *and the pseudorandom function* PRF *used inside the scheme* $\mathsf{R.ECom}$, H_1 *is computationally indistinguishable from* H_2.

Proof. Recall from the construction of the scheme $\mathsf{R.ECom}$, that for every honest party P_i and malicious party P_j, in each thread, in H_1, $\mathsf{r.ec}_{b,3}^{i\to j}$ consists of $(\alpha_b^{i\to j}, \mathsf{brew.ec}_{b,3}^{i\to j}, \beta_b^{i\to j})$ where all 3 are picked uniformly at random and in H_2, $\mathsf{r.ec}_{b,3}^{i\to j}$ consists of $(\alpha_b^{i\to j}, \mathsf{brew.ec}_{b,3}^{i\to j}, \beta_b^{i\to j})$ where $\mathsf{brew.ec}_{b,3}^{i\to j} = \mathsf{BRew.ECom}_3(\mathsf{r}_b^{i\to j}, \mathsf{brew.ec}_{b,1}^{i\to j}, \mathsf{brew.ec}_{b,2}^{j\to i}; \mathsf{r}_{b,\mathsf{brew.ec}})$, $\beta_b^{i\to j} = \mathsf{PRF}(\mathsf{r}_b^{i\to j}, \alpha_b^{i\to j}) \oplus \mathsf{x}_i$ and $\mathsf{r}_b^{i\to j}, \mathsf{r}_{b,\mathsf{brew.ec}}$ are picked uniformly at random. Note that only $\alpha_b^{i\to j}$ is sampled afresh in each thread.

We will now prove this subclaim via a series of intermediate sub-hybrids $\mathsf{Sub.Hyb}_1$ to $\mathsf{Sub.Hyb}_3$ where $\mathsf{Sub.Hyb}_1$ corresponds to H_1 and $\mathsf{Sub.Hyb}_3$ corresponds to H_2.

- $\mathsf{Sub.Hyb}_1$**:** This is same as H_1.
- $\mathsf{Sub.Hyb}_2$**:** In this hybrid, for every honest party P_i and malicious party P_j, pick a random value $\mathsf{r}_b^{i\to j}$ and compute $\mathsf{brew.ec}_{b,3}^{i\to j} = \mathsf{BRew.ECom}_3(\mathsf{r}_b^{i\to j}, \mathsf{brew.ec}_{b,1}^{i\to j}, \mathsf{brew.ec}_{b,2}^{j\to i}; \mathsf{r}_{b,\mathsf{brew.ec}})$. Observe that this is the same across all the threads.
  This is indistinguishable from the previous sub-hybrid by the hiding property of the underlying extractable commitment scheme $\mathsf{BRew.ECom}$.
- $\mathsf{Sub.Hyb}_3$**:** In this hybrid, for every honest party P_i and malicious party P_j, in each thread, compute $\beta_b^{i\to j} = \mathsf{PRF}(\mathsf{r}_b^{i\to j}, \alpha_b^{i\to j}) \oplus \mathsf{x}_i$. This is same as H_2.
  This is indistinguishable from the previous sub-hybrid by the security of the pseudorandom function PRF.

□

Sub-Claim 7. *Assuming the bounded rewinding security of the scheme* RWI, H_2 *is computationally indistinguishable from* H_3.

Proof. The only difference between H_2 and H_3 is that in H_3, in every thread, the simulator now computes the WI using a witness for the second statement. Suppose there exists an adversary $\mathcal{A}$ that can distinguish between the two hybrids. We will use $\mathcal{A}$ to design an adversary $\mathcal{A}_{\mathsf{RWI}}$ that breaks the bounded rewinding security of the scheme RWI.

$\mathcal{A}_{\mathsf{RWI}}$ interacts with a challenger $\mathcal{C}_{\mathsf{RWI}}$. $\mathcal{A}_{\mathsf{RWI}}$ performs the role of $\mathsf{Sim}_{\mathsf{Hyb}}$ in its interaction with $\mathcal{A}$ exactly as done in H_2. $\mathcal{A}_{\mathsf{RWI}}$ picks an honest party P_i and a malicious party P_j uniformly at random. Initially, it receives a message rwi_1 from $\mathcal{C}_{\mathsf{RWI}}$ which it sets as the value $\mathsf{rwi}_1^{i\to j}$ in its interaction with $\mathcal{A}$ in round 1. Then, $\mathcal{A}_{\mathsf{RWI}}$ creates two sets of 3 look-ahead threads each. For each thread, on receiving $\mathsf{rwi}_2^{j\to i}$ in round 2, $\mathcal{A}_{\mathsf{RWI}}$ forwards this to $\mathcal{C}_{\mathsf{RWI}}$ as its first round message. For each thread, $\mathcal{A}_{\mathsf{RWI}}$ also sends the statement $\mathsf{st}_2^{i\to j} =$
$(\mathsf{r.ec}_{a,1}^{i\to j}, \mathsf{r.ec}_{a,2}^{j\to i}, \mathsf{r.ec}_{a,3}^{i\to j}, \mathsf{r.ec}_{b,1}^{i\to j}, \mathsf{r.ec}_{b,2}^{j\to i}, \mathsf{r.ec}_{b,3}^{i\to j}, \mathsf{msg}_{2,i}, \mathsf{Trans}_1, \mathsf{c}_1^{i\to j}, \mathsf{c}_2^{j\to i}, \mathsf{c}_3^{i\to j}, \mathsf{td}_1^{j\to i}, \mathsf{nc}_i) \in L_2^{i\to j}$ where the other values are generated as in H_2.

$\mathcal{A}_{\mathsf{RWI}}$ also sends the pair of witnesses $(\mathsf{x}_i, \mathsf{r}_i, \mathsf{r}_{a,\mathsf{r.ec}}^{i\to j}, \bot, \bot, \bot, \bot)$ and $(\mathsf{x}_i, \mathsf{r}_i, \bot, \mathsf{r}_{b,\mathsf{r.ec}}^{i\to j}, \bot, \bot, \bot)$ for the main thread and each of the first set of look-ahead threads. For each look-ahead thread in the second set, $\mathcal{A}_{\mathsf{RWI}}$ sends the pair of witnesses $(0, \mathsf{r}_i, \mathsf{r}_{a,\mathsf{r.ec}}^{i\to j}, \bot, \bot, \bot, \bot)$ and $(0, \mathsf{r}_i, \bot, \mathsf{r}_{b,\mathsf{r.ec}}^{i\to j}, \bot, \bot, \bot)$ For each thread, $\mathcal{A}_{\mathsf{RWI}}$ receives a third round message rwi_3 which is set as $\mathsf{rwi}_3^{i\to j}$ in its interaction with $\mathcal{A}$ in round 3 of protocol π. The rest of protocol π is performed exactly as in H_2. Observe that similar to the proof of Claim 12, given just 3 look-ahead threads in each set, we will still be able to simulate the weak ZK with noticeable probability.

Observe that the first case corresponds to H_2 while the second case corresponds to H_3. Therefore, if the adversary $\mathcal{A}$ can distinguish between these two hybrids, $\mathcal{A}_{\mathsf{RWI}}$ will use the same guess to break the rewinding security of the scheme RWI which is a contradiction because we assumed that RWI is secure even in the presence of 6 rewound look-ahead threads (we set L to be 6). □

Sub-Claim 8. *Assuming the hiding property of the non-interactive commitment scheme* NCom*,* H_3 *is computationally indistinguishable from* H_3*.*

Proof. The only difference between the two hybrids is that in H_3, for every honest party P_i, nc_i is computed as a commitment of 1 in the main thread while in H_4 it is computed as a commitment of 0. Note that in both hybrids, there is no change on any of the look-ahead threads. Thus, if there exists an adversary that can distinguish between the two hybrids, there exists a reduction that can break the hiding property of the commitment scheme. □

Sub-Claim 9. *Assuming the bounded rewinding security of the scheme* RWI*,* H_4 *is computationally indistinguishable from* H_5*.*

Proof. The proof is very similar to the proof of Sub-Claim 7. The only difference being that here, the witness is changed only in the main thread and not in all the threads. □

Sub-Claim 10. *Assuming the security of the extractable commitment* $\mathsf{BRew.ECom}$ *and the pseudorandom function* PRF *used inside the scheme* $\mathsf{R.ECom}$*,* H_5 *is computationally indistinguishable from* H_6*.*

Proof. The proof is similar to the proof of Sub-Claim 6.

□

Sub-Claim 11. *Assuming the security of the semi-malicious MPC protocol* π^{SM}*,* H_6 *is computationally indistinguishable from* H_7*.*

Proof. The only difference between H_6 and H_7 is that in H_7, in the main thread, the simulator now computes the second message of protocol π^{SM} using the simulated algorithms $\mathcal{S}_2$. Assume for the sake of contradiction that there exists an adversary $\mathcal{A}$ that can distinguish between the two hybrids. We will use $\mathcal{A}$ to design an adversary $\mathcal{A}_{\pi^{\mathsf{SM}}}$ that breaks the security of the protocol π^{SM}.

$\mathcal{A}_{\pi^{\mathsf{SM}}}$ performs the role of $\mathsf{Sim}_{\mathsf{Hyb}}$ in its interaction with $\mathcal{A}$ exactly as done in H_6. $\mathcal{A}_{\pi^{\mathsf{SM}}}$ also interacts with a challenger $\mathcal{C}_{\pi^{\mathsf{SM}}}$ and corrupts the same parties as done by $\mathcal{A}$. For every honest party P_i, $\mathcal{A}_{\pi^{\mathsf{SM}}}$ receives a first round message $\mathsf{msg}_{1,i}$ which is sent to $\mathcal{A}$ in round 1 of protocol π on the main thread. On receiving $\mathsf{msg}_{1,j}$ for every malicious party P_j in round 1 of the main thread from $\mathcal{A}$, $\mathcal{A}_{\pi^{\mathsf{SM}}}$ forwards this to $\mathcal{C}_{\pi^{\mathsf{SM}}}$ as the first round messages of the malicious parties. Similarly, the messages $\mathsf{msg}_{2,i}$ and $\mathsf{msg}_{2,j}$ corresponding to every honest party P_i and malicious party P_j are sent across between $\mathcal{C}_{\pi^{\mathsf{SM}}}$ and $\mathcal{A}$ via $\mathcal{A}_{\pi^{\mathsf{SM}}}$ in round 3 of protocol π on the main thread. The rest of protocol π is performed exactly as in H_6. In particular, $\mathcal{A}_{\pi^{\mathsf{SM}}}$ generates the messages of protocol π^{SM} in the look-ahead threads on its own even though the first message of the protocol was received externally from $\mathcal{C}_{\pi^{\mathsf{SM}}}$. Recall that this follows from a property of the protocol π^{SM} that the first round message is independent of the input and doesn't have any secret information used to generate the second round messages.

Observe that when $\mathcal{C}_{\pi^{\mathsf{SM}}}$ computes the messages of protocol π^{SM} honestly, $\mathcal{A}$'s view corresponds to H_6 and when $\mathcal{C}_{\pi^{\mathsf{SM}}}$ computes simulated messages, $\mathcal{A}$'s view corresponds to H_7. Therefore, if $\mathcal{A}$ can distinguish between these two hybrids, $\mathcal{A}_{\pi^{\mathsf{SM}}}$ will use the same distinguishing guess to break the security of protocol π^{SM}.

□

Sub-Claim 12. *Assuming the bounded rewinding security of the scheme* RWI, H_7 *is computationally indistinguishable from* H_8.

Proof. The proof is same as the proof of Sub-Claim 7 discussed above. □

Sub-Claim 13. *Assuming the security of the extractable commitment* $\mathsf{BRew.ECom}$ *and the pseudorandom function* PRF *used inside the scheme* $\mathsf{R.ECom}$, H_8 *is computationally indistinguishable from* H_9.

Proof. The proof is same as the proof of Sub-Claim 6 discussed above. □

Sub-Claim 14. *Assuming the bounded rewinding security of the scheme* RWI, H_9 *is computationally indistinguishable from* H_{10}.

Proof. The proof is same as the proof of Sub-Claim 9 discussed above. □

Sub-Claim 15. *Assuming the hiding property of the non-interactive commitment scheme* NCom, H_{10} *is computationally indistinguishable from* H_{11}.

Proof. The proof is same as the proof of Sub-Claim 8 discussed above. □

Sub-Claim 16. *Assuming the Weak Zero Knowledge property of the argument system* WZK, H_{11} *is computationally indistinguishable from* H_{12}.

Proof. The proof is same as the proof of Sub-Claim 5 discussed above. □