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 how to realize promise ZK 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.
- 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 LWE.

Previously, four round MPC protocols required sub-exponential-time hardness assumptions and no multi-party three-round protocols were known for any relaxed security notions with polynomial-time simulation 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. 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 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. We call this primitive a *distributional delayed-input promise zero-knowledge argument*. 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 delayed-input promise ZK argument system based on injective one-way functions.

Theorem 1 (Informal)**.** *Assuming injective one-way functions, there exists a three round distributional delayed-input 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 that converts any three-round semi-malicious MPC protocol, where the first round is public-coin (i.e., the honest parties simply send randomness in the first round), into 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 enhanced trapdoor permutations and public-key encryptions schemes that admit affine homomorphisms with equivocation (which in turn can be based on LWE/DDH/QR; see [HHPV17]). 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.

[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].

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. 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.

[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.

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 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

[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.

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 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-

[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.

round non-malleable commitments [GPR16, Khu17] to achieve *simulation soundness* [Sah99] in 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

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

probability, it can extract *both* the trapdoor and the inputs from the adversary. In the first 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 7.

Then, in Section 5, 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 6.

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 Secure Multiparty Computation

In this work we follow the standard real/ideal world paradigm for defining secure multi-party computation. We refer the reader to [Gol04] for the precise definition.

Semi-malicious adversary. An adversary is said to be semi-malicious if it follows the protocol correctly, but with potentially maliciously chosen randomness.

3.2 Delayed-Input Interactive Arguments

In this section, we describe delayed-input interactive arguments.

Definition 1 (Delayed-Input Interactive Arguments)**.** *An n-round delayed-input interactive protocol (P, V) for deciding a language L is an argument system for L that satisfies the following properties:*

- **Delayed-Input Completeness.** *For every security parameter $\lambda \in \mathbb{N}$, and any $(x, w) \in R_L$ such that $|x| \leq 2^\lambda$,*

$$\Pr[(P, V)(1^\lambda, x, w) = 1] = 1 - \mathsf{negl}(\lambda).$$

where the probability is over the randomness of P and V. Moreover, the prover's algorithm initially takes as input only 1^λ, and the pair (x, w) is given to P only in the beginning of the n'th round.

- **Delayed-Input Soundness.** *For any PPT cheating prover P^* that chooses x^* (adaptively)* after *the first $n-1$ messages, it holds that if $x^* \notin L$ then*

$$\Pr[(P^*, V)(1^\lambda, x^*) = 1] = \mathsf{negl}(\lambda).$$

where the probability is over the random coins of V.

Remark 1. *We note that in a delayed-input interactive argument satisfying Definition 1, completeness and soundness also hold when (part of) the instance is available in the first $(n-1)$ rounds.*

We will also consider delayed-input interactive arguments in the simultaneous-message setting, that satisfy soundness against rushing adversaries.

3.3 Extractable Commitments

Definition 2 (Extractable Commitments with Over-extraction)**.** *Consider any statistically binding, computationally hiding commitment scheme $\langle C, R\rangle$, and let $\mathsf{Com}(m; r)$ denote the algorithm that on input m and randomness r outputs the transcript of the commitment phase. Then $\langle C, R\rangle$ is said to be extractable if there exists an expected PPT oracle algorithm $\mathcal{E}$, such that for any PPT cheating committer C^* the following holds. Let $\mathsf{Trans}\langle C^*, R\rangle$ denote a transcript of the interaction between C^* and R. Then $\mathcal{E}^{C^*}(\mathsf{Trans}\langle C^*, R\rangle)$ outputs m, r such that over the randomness of $\mathcal{E}$ and of sampling $\mathsf{Trans}\langle C^*, R\rangle$:*

$$\Pr[\mathsf{Com}(m; r) = (\mathsf{Trans}\langle C^*, R\rangle) \bigvee \nexists(\widetilde{m}, \widetilde{r}) \text{ such that } \mathsf{Trans}\langle C^*, R\rangle = \mathsf{Com}(\widetilde{m}; \widetilde{r})] = 1 - \mathsf{negl}(\lambda).$$

Definition 3 (k-Extractable Commitments with Over-extraction)**.** *An extractable commitment with over-extraction is said to be k-extractable if there exists a polynomial $p(\cdot)$ such that the extractor $\mathcal{E}^{C^*}(\mathsf{Trans}\langle C^*, R\rangle)$* with $k-1$ queries to C^*, *outputs m, r such that over the randomness of $\mathcal{E}$ and of sampling $\mathsf{Trans}\langle C^*, R\rangle$:*

$$\Pr[\mathsf{com}(m; r) = (\mathsf{Trans}\langle C^*, R\rangle) \bigvee \nexists(\widetilde{m}, \widetilde{r}) \text{ such that } \mathsf{Trans}\langle C^*, R\rangle = \mathsf{com}(\widetilde{m}; \widetilde{r})] \geq \frac{1}{p(\lambda)}.$$

Delayed-Input Extractable Commitments. We say that an extractable commitment is *delayed-input* if the committer uses the input message m only in the last round of the protocol.

Three round delayed-input extractable commitments that satisfy K-extractability can be constructed from injective one-way functions.

3.4 Non-Malleable Commitments

We start with 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]).

In the real experiment, a man-in-the-middle adversary MIM interacts with a committer C in the left session, and with a receiver R in the right session. Without loss of generality, we assume that each session has identities or tags, and require non-malleability only when the tag for the left session is different from the tag for the right session.

At the start of the experiment, the committer C receives an input v and MIM receives an auxiliary input z, which might contain a priori information about v. Let $\mathsf{MIM}_{\langle C,R\rangle}(\mathsf{val}, z)$ be a random variable that describes the value $\widetilde{\mathsf{val}}$ committed by MIM in the right session, jointly with the view of MIM in the real experiment.

In the ideal experiment, a PPT simulator $\mathcal{S}$ directly interacts with 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 $\mathcal{S}$.

In either of the two experiments, if the tags in the left and right interaction are equal, then the value $\widetilde{\mathsf{val}}$ committed in the right interaction, is defined to be $\perp$.

We define a strengthened version of non-malleable commitments for use in this paper.

Definition 4 (Special Non-malleable Commitments)**.** *A three round commitment scheme $\langle C, R\rangle$ is said to be special non-malleable if:*

- *For every* synchronizing[10] *PPT* MIM*, there exists a PPT simulator $\mathcal{S}$ such that the following ensembles are computationally indistinguishable:*

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

- *$\langle C, R\rangle$ is delayed-input, that is, correctness holds even when the committer obtains his input only in the last round.*

- *$\langle C, R\rangle$ satisfies* last-message pseudorandomness*, that is, for every non-uniform PPT receiver R^*, it holds that $\{\mathsf{REAL}_0^{R^*}(1^\lambda)\}_\lambda$ and $\{\mathsf{REAL}_1^{R^*}(1^\lambda)\}_\lambda$ are computationally indistinguishable, where for $b \in \{0, 1\}$, the random variable $\mathsf{REAL}_b^{R^*}(1^\lambda)$ is defined via the following experiment.*

  1. *Run $C(1^\lambda)$ and denote its output by (com_1, σ), where σ is its secret state, and com_1 is the message to be sent to the receiver.*
  2. *Run the receiver $R^*(1^\lambda, \mathsf{com}_1)$, who outputs x and a message com_2.*
  3. *If $b = 0$, run $C(\sigma, \mathsf{com}_2, x)$ and send its message com_3 to R^*. Otherwise, if $b = 1$, compute $\mathsf{com}_3 \xleftarrow{\$} \{0,1\}^m$ and send it to R^*. Here $m = m(\lambda)$ denotes $|\mathsf{com}_3|$.*
  4. *The output of the experiment is the output of R^*.*

- *$\langle C, R\rangle$ satisfies 2-extractability according to Definition 3.*

Goyal et al. [GPR16] construct three-round special non-malleable commitments satisfying Definition 4 based on injective OWFs.

Imported Theorem 1 ([GPR16])**.** *Assuming injective one-way functions, there exists a three round non-malleable commitment satisfying Definition 4.*

4 Building Blocks

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

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

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 a valid trapdoor, and (ii) Extraction, i.e., there exists an expected PPT algorithm (a.k.a. extractor) that can extract a trapdoor from an adversarial sender via rewinding.

We construct a three-round trapdoor generation protocol where the first message sent by the sender determines the set of valid trapdoors, and in the next two rounds the sender proves that indeed it knows a valid trapdoor. Such schemes 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 based on one-way functions. Our construction is a minor variant of the trapdoor generation protocol from [COSV17a].

Syntax. A trapdoor generation protocol

$$\mathsf{TDGen} = (\mathsf{TDGen}_1, \mathsf{TDGen}_2, \mathsf{TDGen}_3, \mathsf{TDOut}, \mathsf{TDValid}, \mathsf{TDExt})$$

is a three round protocol between two parties - a sender (trapdoor generator) S and receiver R that proceeds as below.

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

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

Extraction. There exists a PPT extractor algorithm TDExt that, given a set of values[11] $(\mathsf{td}_1, \{\mathsf{td}_2^i, \mathsf{td}_3^i\}_{i=1}^3)$ such that $\mathsf{td}_2^1, \mathsf{td}_2^2, \mathsf{td}_2^3$ are distinct and $\mathsf{TDOut}(\mathsf{td}_1, \mathsf{td}_2^i, \mathsf{td}_3^i) = 1$ for all $i \in [3]$, outputs a trapdoor t such that $\mathsf{TDValid}(\mathsf{t}, \mathsf{td}_1) = 1$.

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

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 any (possibly cheating) 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 5 (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,*

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

where the probability is over the random coins of S*, and where* 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 describe a three round trapdoor generation protocol based on one way functions.

Let S and R denote the sender and the receiver, respectively. Let λ denote the security parameter. Let $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme that is existentially unforgeable against chosen-message attacks. Such schemes are known based on one-way functions.

Theorem 5. *Assuming the existence of one way functions, the protocol* π^{TD} *described in Figure 1 is a 1-rewinding secure 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 $\mathcal{C}_{\mathsf{Sign}}$. Upon receiving a verification key vk from $\mathcal{A}_{\mathsf{Sign}}$, it sets to td_1 and sends it to R^* in round 1.

Upon 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, upon 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 $\mathsf{m}_1, \mathsf{m}_2, \mathsf{m}_3$ are distinct and $V(\mathsf{vk}, \mathsf{m}_i, \sigma_i) = 1$ 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 $\mathcal{C}_{\mathsf{Sign}}$ 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 $\{\mathsf{m}_i, \sigma_i\}_{i=1}^3$ such that m_i are all distinct and $\mathsf{Verify}(\mathsf{vk}, \mathsf{m}_i, \sigma_i) = 1$ for every $i \in [3]$. 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.

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 $\mathsf{m}_1, \mathsf{m}_2, \mathsf{m}_3$ are distinct and $V(\mathsf{td}_1, \mathsf{m}_i, \sigma_i) = 1$ for all $i \in [3]$.

Figure 1: Trapdoor Generation Protocol.

4.2 WI with Bounded Rewinding Security

We define the notion of three-round delayed-input witness indistinguishable (WI) 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 the literature previously in the non-delayed-input setting. For example, Goyal et al. [GRRV14] construct and use the notion of WI arguments with 1-rewinding security based on injective one-way functions.

We formally define three-round delayed-input WI with bounded-rewinding security here. In appendix Appendix A, we describe a construction for the same which is obtained by combining a delayed-input WI [LS90] that is not rewinding secure with a non-delayed-input WI that is bounded-rewinding secure [IKOS07]. For our applications, we instantiate the rewinding parameter B with the value 5.

Definition 6 (3-Round Delayed-Input WI with Bounded Rewinding Security)**.** *Fix a positive integer B. A delayed-input 3-round interactive argument (as defined in Definition 1) for an NP language L, with an NP relation R_L is said to be WI with B-*Rewinding Security *if for every non-uniform PPT interactive Turing Machine V^*, it holds that $\{\mathsf{REAL}_0^{V^*}(1^\lambda)\}_\lambda$ and $\{\mathsf{REAL}_1^{V^*}(1^\lambda)\}_\lambda$ are computationally indistinguishable, where for $b \in \{0,1\}$ the random variable $\mathsf{REAL}_b^{V^*}(1^\lambda)$ is defined via the following experiment. In what follows we denote by P_1 the prover's algorithm in the first round, and similarly we denote by P_3 his algorithm in the third round.*

Experiment $\mathsf{REAL}_b^{V^*}(1^\lambda)$**:**

1. *Run $P_1(1^\lambda)$ and denote its output by (rwi_1, σ), where σ is its secret state, and rwi_1 is the message to be sent to the verifier.*

2. *Run the verifier* $V^*(1^\lambda, \mathsf{rwi}_1)$*, who outputs* (x, w_0, w_1) *and a message* rwi_2*.*

3. *Run* $P_3(\sigma, \mathsf{rwi}_2, x, w_b)$*, where* P_3 *is the (honest) prover's algorithm for generating the third message of the WI protocol, and send its message* rwi_3 *to* V^**.*

4. *Set a counter* $i = 0$*.*

5. *If* $i < B$*, then set* $i = i + 1$*, and* V^* *(given all the information so far) generates another message* rwi_2^i*, and receives the (honest) prover's message* $P_3(\sigma, \mathsf{rwi}_2^i, x, w_b)$*. Repeat this step until* $i = B$*.*

6. *The output of the experiment is the output of* V^**.*

In Appendix A, we prove the following theorem:

Theorem 6. *Assuming injective one-way functions, there exists a three round delayed-input witness-indistinguishable argument system with* $(B = 5)$*-rewinding security.*

5 Promise Zero Knowledge

In this section, we introduce our 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 "valid" verifiers (adversaries that do not send invalid messages).

Validity Check. Before defining promise ZK, we enhance the syntax for simultaneous-message interactive arguments to include an additional algorithm Valid. That is, a simultaneous-message interactive argument is denoted by (P, V, 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" message in every round (including the last round). We do this by using the Valid algorithm.

This algorithm Valid is protocol specific. For example, if the honest verifier is instructed to prove knowledge of a trapdoor that he generated, and the proof fails, then his messages are not valid. Importantly, even if only the verifier's last message is invalid, and even though prover does not need to explicitly respond to this message[12] we refer to this transcript as invalid. We denote by Valid the (public verification) algorithm which checks whether the transcript, including the verifier's last message, is valid or not, that is,

$$\mathsf{Valid}(\mathsf{Trans}(P(x, w), V^*)) = 1$$

if and only if all the messages sent by V^* appear to be valid, given the transcript. The correctness requirement of this algorithm is that for messages generated by verifier V honestly according to the protocol,

$$\Pr[\mathsf{Valid}(\mathsf{Trans}(P(x, w), V)) = 1] = 1.$$

Looking ahead, in our protocols, at the end of each execution of the ZK protocol, the prover will check whether the verifier sent "valid" messages, and if not, the prover will abort.

[12] We use this promise ZK protocol as a building block in our MPC protocols, and in these protocols, the party acting as prover does indeed read this last ZK message sent by the verifier, and based on its validity decides whether to abort the MPC protocol. See, for example, Section 6.

5.1 Definitions

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, the zero knowledge property is only defined w.r.t. a special class of verifiers who generate a valid transcript, with some noticeable probability. In order to define this notion, we need to have an estimation of the probability that the cheating verifier sends an invalid message throughout the protocol.

Validity Approximation. Consider a delayed-input simultaneous message interactive argument system (P, V, Valid). Consider any verifier V^*, and any efficiently sampleable distribution $\mathcal{D} = \{\mathcal{D}_\lambda\}$, where $\mathcal{D}_\lambda$ samples pairs (x, w) such that $x \in \{0,1\}^\lambda$ and $(x, w) \in R_L$

In what follows we denote by $P = (P_1, P_2)$, a prover that is split into two parts. First, $(\mathsf{view}_{V^*,1}, \mathsf{st}) \leftarrow P_1(1^\lambda)$ is obtained, and then $P_2(x, w, \mathsf{st})$ continues the rest of the P algorithm with V^*. This is done primarily because we would like to approximate the the validity probability of the adversary conditioned on fixed first $\mathsf{view}_{V^*,1}$.

Let $\mathsf{Trans}(P_2(x, w, \mathsf{st}), V^*)$ denote the protocol transcript between P_2 and V^*: that is, $\mathsf{Trans}(P, V^*) = (\mathsf{view}_{V^*,1}, \mathsf{Trans}(P_2(x, w, \mathsf{st}), V^*))$. Let

$$q_{\mathsf{view}_{V^*,1}} = \Pr[\mathsf{Valid}(\mathsf{view}_{V^*,1}, \mathsf{Trans}(P_2(x, w, \mathsf{st}), V^*)) = 1 | (\mathsf{view}_{V^*,1}, \cdot) \leftarrow P_1(1^\lambda)]$$

where the probability is over the generation of $(x, w) \leftarrow \mathcal{D}_\lambda$ and the coins of P_2. We emphasize that $q_{\mathsf{view}_{V^*,1}}$ depends on $\mathcal{D}$ and on V^*, we omit this dependence from the notation to avoid cluttering.

Definition 7. *A PPT oracle algorithm* $\mathsf{pExtract}_c$ *is said to be a* validity approximation algorithm, *if the following holds for all malicious verifiers V^* and for all efficiently sampleable distributions $\mathcal{D} = \{\mathcal{D}_\lambda\}$:*

- *If* $\mathsf{pExtract}_c^{V^*,\mathcal{D}}(\mathsf{view}_{V^*,1}, \mathsf{st}) = 0$, *then* $q_{\mathsf{view}_{V^*,1}} < 2 \cdot \lambda^{-c}$.
- *Otherwise, if* $\mathsf{pExtract}_c^{V^*,\mathcal{D}}(\mathsf{view}_{V^*,1}, \mathsf{st}) = p$, *then* $p \geq \lambda^{-c}$ *and* $\frac{p}{2} < q_{\mathsf{view}_{V^*,1}} < 2 \cdot p$.

We now formalize our notion of promise ZK. We note that this only considers the delayed-input distributional setting. For simplicity of exposition, we restrict ourselves to 3 round protocols since this work is only concerned with constructions and applications of 3 round promise zero-knowledge. We note that this definition can be extended naturally to any number of rounds.

Definition 8 (Promise Zero Knowledge)**.** *A 3-round distributional delayed-input simultaneous-message interactive argument* (P, V, Valid) *for a language L is said to be* promise zero knowledge against delayed-input verifiers *if there exists an oracle machine* $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3)$ *such that for every constant $c > 0$, and any validity approximation algorithn* $\mathsf{pExtract}_c$, *for every polynomials $\nu = \nu(\lambda)$ and $\widetilde{\nu} = \widetilde{\nu}(\lambda)$, for every efficiently sampleable distribution $\mathcal{D} = \{\mathcal{D}_\lambda\}$ such that* $\mathsf{Supp}(\mathcal{D}_\lambda) = \{(x, w) : x \in L \cap \{0,1\}^\lambda, w \in R_L(x)$ *where* $x = (x_2, x_3), w = (w_2, w_3)\}$, *for any delayed-input PPT verifier V^* that obtains x_i in round i and any $z \in \{0,1\}^{\mathsf{poly}(\lambda)}$, conditions 1 and 2 are true for* REAL_{V^*} *and* IDEAL_{V^*} *defined below. Here* view_i *denotes a partial view of V^* until the end of the i^{th} round.*

- REAL_{V^*} *is computed as follows:*
  - *Sample* $(\mathsf{view}_{V^*,1}, \mathsf{st}) \leftarrow P_1(1^\lambda)$.
  - *Sample* $(x, w) \leftarrow \mathcal{D}_\lambda$ *where* $x = (x_2, x_3)$.

- *Execute the interaction* $(\mathsf{view}_{V^*,1}, \langle P_2(x, w, \mathsf{st}), V^* \rangle)$*, where* V^* *obtains* x_i *in round* i*.*
- *The output of the experiment is* $(x, \langle P(x, w), V^* \rangle)$*.*

- IDEAL_{V^*} *is computed as follows:*
  - *Sample* $(\mathsf{view}_{V^*,1}, \mathsf{st}) \leftarrow P_1(1^\lambda)$*.*
  - *Compute* $p = \mathsf{pExtract}_c^{V^*}(\mathsf{view}_{V^*,1}, \mathsf{st})$*.*
  - *Sample* $(x, w) \leftarrow \mathcal{D}_\lambda$ *where* $x = (x_2, x_3)$*.*
  - *If* $p = 0$*,*
    - *Execute the interaction* $(\mathsf{view}_{V^*,1}, \langle P_2(x, w, \mathsf{st}), V^* \rangle)$*, where* V^* *obtains* x_i *in round* i*.*
    - *The output of the experiment is* $(x, \langle P(x, w), V^* \rangle)$*.*
  - *Else, execute* $\mathsf{Sim}^{V^*}(x, \mathsf{view}_{V^*,1}, \mathsf{st}, p) \to (\mathsf{view}_{V,2}, \mathsf{view}_{V,3})$*, which operates as follows:*
    - $\mathsf{Sim}_1^{V^*}(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}, p) \to \mathsf{st}_1$*.*
    - *The compute* $\mathsf{Sim}_2^{V^*}(x_2, \mathsf{view}_{V^*,1}, \mathsf{st}_1) \to (\mathsf{view}_{V^*,2}, \mathsf{st}_2)$*.*
    - *Finally, compute* $\mathsf{Sim}_3^{V^*}(x_3, \mathsf{view}_{V^*,1}, \mathsf{view}_{V^*,2}, \mathsf{st}_2)$ *to output* $(\mathsf{view}_{V^*,3})$*.*

Then, conditions 1 and 2 are as follows:

1. *No PPT distinguisher can distinguish* REAL_{V^*} *from* IDEAL_{V^*} *with advantage greater than* λ^{-c}*.*
2. *For any input* $x = (x_2, x_3)$*, the running time of* $\mathsf{Sim}_1^{V^*}(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}, p)$ *is polynomial in* λ *and linear in* $\frac{1}{p}$*, and the running times of* $\mathsf{Sim}_2^{V^*}(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{st})$ *and* $\mathsf{Sim}_3^{V^*}(x_2, x_3, \mathsf{view}_{\mathsf{MIM},1}, \mathsf{view}_{\mathsf{MIM},2}, \mathsf{st}_2)$ *are polynomial in* λ *and independent of* p*.*

Going forward, we use *promise ZK argument* to refer to a distributional promise zero-knowledge simultaneous-message argument system, satisfying delayed-input completeness and soundness, as well as zero-knowledge against delayed-input verifiers according to Definition 8.

Defining Simulation-Sound Promise ZK in the multi-party setting. We now consider a man-in-the-middle adversary that interacts in promise zero-knowledge protocols as follows: It opens polynomially many sessions where it plays the role of verifier interacting with an honest prover; these are called "left" sessions, and we denote by ν the number of such left sessions. We note that in all left sessions, the honest prover proves the same statement with the same witness. It can simultaneously initiate polynomially many sessions where it plays the role of prover interacting with an honest verifier: these are called "right" sessions, and we denote by $\widetilde{\nu}$ the number of such right sessions. We restrict ourselves to *synchronous* (rushing) adversaries, that for each round j, send all their j'th round messages (in all sessions), before observing any of the honest parties messages for the next round of the protocol.

We formalize the notion of simulation-soundness against a rushing man-in-the-middle adversary below, where we use $\widetilde{a}$ to denote any random variable a that corresponds to a right session.

Redefining Validity Approximation. Similarly to before, we need to approximate the probability that the messages sent by a man-in-the-middle adversary in the left execution are valid, conditioned on all messages in the first round of the protocol. We consider ν "left" sessions and $\widetilde{\nu}$ "right" sessions. Similar to the setting of promise ZK, we denote by $P = (P_1, P_2)$, an honest prover for the "left" sessions that is split into two parts, P_1 generates the first round message, and P_2 generates the messages of the second and third rounds. Below, we abuse notation and use P, P_1, P_2 not only to denote the interaction of the honest prover in a single session, but also to denote the interaction of the honest prover in all ν left sessions, using independent randomness for each such execution. Let $\mathsf{Trans}_{\mathrm{left}}(P_2(x, w, \mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}), \mathsf{MIM})$ denote all the transcripts in the "left" sessions between $P_2(x, w, \mathsf{view}_{\mathsf{MIM},1}, \mathsf{st})$ and MIM, which can be decomposed as follows: $\mathsf{Trans}_{\mathrm{left}}(P, \mathsf{MIM}) = (\mathsf{view}_{\mathsf{MIM},1}, \mathsf{Trans}_{\mathrm{left}}(P_2(x, w, \mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}), \mathsf{MIM}))$. For any $\mathsf{view}_{\mathsf{MIM},1}$ sampled according to honest prover and verifier strategy as described above, let

$$q_{\mathsf{view}_{\mathsf{MIM},1}} = \Pr[\mathsf{Valid}(\mathsf{Trans}_{\mathrm{left}}(P_2(x, w, \mathsf{st}))) = 1]$$

where Valid above refers to the AND of all the validity tests for each of the ν left sessions, and the probability is over the generation of $(x, w) \leftarrow \mathcal{D}_\lambda$ and the coins of each of the ν instantiations of P_2. We emphasize that $q_{\mathsf{view}_{\mathsf{MIM},1}}$ depends on $\mathcal{D}$ and on MIM, we omit this dependence from the notation to avoid cluttering. We re-define the algorithm $\mathsf{pExtract}_c$ from Definition 8 to depend additionally on the honest verifier first messages in the right sessions.

Definition 9. *A PPT oracle algorithm* $\mathsf{pExtract}_c$ *is said to be a* validity approximation algorithm*, if the following holds for all* MIM *and for all efficiently sampleable distributions* $\mathcal{D} = \{\mathcal{D}_\lambda\}$*, with probability at least* $1 - 2^{-\lambda}$ *over the coins of the algorithm, we have that:*

- *If* $\mathsf{pExtract}_c^{\mathsf{MIM},\mathcal{D}}(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}) = 0$*, then* $q_{\mathsf{view}_{\mathsf{MIM},1}} < 2 \cdot \lambda^{-c}$*.*
- *Else, if* $\mathsf{pExtract}_c^{\mathsf{MIM},\mathcal{D}}(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}) = p$*, then* $p \geq \lambda^{-c}$ *and* $p < 2q_{\mathsf{view}_{\mathsf{MIM},1}}$*.*

Remark 2. *We briefly describe a canonical polynomial-time validity approximation algorithm for any constant* $c > 0$ *here:*

1. $\mathsf{pExtract}_c^{\mathsf{MIM},\mathcal{D}}(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{st})$ *executes* $\lambda^2 \cdot \lambda^c$ *independent executions of all sessions with* MIM*, using freshly sampled instance-witness pairs from the distribution* $\mathcal{D}_\lambda$ *to complete the left executions in the role of the honest provers, and acting as honest verifiers in the right sessions.*
2. *Let* ρ *be the number of these executions that resulted in* all *left executions begin valid. We call such executions successful trials.*
3. *If* $\rho < \lambda^2$*, output* 0*.*
4. *Otherwise, output* $\rho/(\lambda^2 \cdot \lambda^c)$*.*

We now informally analyze this algorithm:

- *Observe that if* $\mathsf{pExtract}_c^{\mathsf{MIM},\mathcal{D}}(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{st})$ *outputs zero, this means that fewer than* λ^2 *trials succeeded. On the other hand, if* $q_{\mathsf{view}_{\mathsf{MIM},1}} \geq 2 \cdot \lambda^{-c}$*, then the expected number of successful trials is at least* $2\lambda^2$*. By a Chernoff bound, except with probability at most* $2^{-\lambda}$*, at least* λ^2 *trials must succeed if* $q_{\mathsf{view}_{\mathsf{MIM},1}} \geq 2 \cdot \lambda^{-c}$*. Thus, the first condition is satisfied.*
- *Observe that if* $\mathsf{pExtract}_c^{\mathsf{MIM},\mathcal{D}}(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{st})$ *outputs a nonzero value, then this value must be at least* λ^{-c} *by construction. And again, the required condition on* $q_{\mathsf{view}_{\mathsf{MIM},1}}$ *follows immediately from a Chernoff bound.*

For simplicity, we restrict ourselves to 3 rounds in the definition below. This suffices for our construction and applications.

Definition 10 (Simulation-Sound Promise Zero Knowledge)**.** *A 3-round publicly-verifiable promise zero-knowledge argument against delayed-input verifiers* (P, V, Valid) *is said to be simulation-sound if there exists an oracle machine* $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3)$ *such that, for every constant* $c > 0$*, and any validity approximation algorithm* $\mathsf{pExtract}_c$*, for every polynomials* $\nu = \nu(\lambda)$ *and* $\widetilde{\nu} = \widetilde{\nu}(\lambda)$*, for every efficiently sampleable distribution* $\mathcal{D} = \{(\mathcal{X}_\lambda, \mathcal{W}_\lambda)\}$ *such that* $\mathsf{Supp}((\mathcal{X}, \mathcal{W})_\lambda) = \{(x, w) : x \in L \cap \{0,1\}^\lambda, w \in R_L(x) \text{ where } x = (x_2, x_3)\}$*, and every distribution* $\mathcal{X}'_\lambda$ *such that* $\mathcal{X}_\lambda$ *and* $\mathcal{X}'_\lambda$ *are computationally indistinguishable, for any PPT synchronous* MIM *that initiates* ν *"left" sessions and* $\widetilde{\nu}$ *"right" sessions, we require the following to hold. Let*

$$\mathsf{Sim}^{\mathsf{MIM}}(x', \mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}, p) \to (\mathsf{view}_{\mathsf{MIM},2}, \mathsf{view}_{\mathsf{MIM},3}, \{\widetilde{x}_i\}_{i \in [\widetilde{\nu}]})$$

where $\mathsf{view}_{\mathsf{MIM},1}$ *are all the messages sent in the first round of the execution with* MIM*, and* st *denotes all the corresponding secret states of the honest parties, and* $p = \mathsf{pExtract}_c^{\mathsf{MIM},\mathcal{D}}(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{st})$*.*

- *For any input* $x' = (x'_2, x'_3)$*, we have that* $\mathsf{Sim}^{\mathsf{MIM}}(x', \mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}, p)$ *operates by first computing*

  $$\mathsf{Sim}_1^{\mathsf{MIM}}(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}, p) \to \mathsf{st}_1$$

  then computing

  $$\mathsf{Sim}_2^{\mathsf{MIM}}(x'_2, \mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}_1) \to (\mathsf{view}_{\mathsf{MIM},2}, \mathsf{st}_2)$$

  and then computing

  $$\mathsf{Sim}_3^{\mathsf{MIM}}(x'_2, x'_3, \mathsf{view}_{\mathsf{MIM},1}, \mathsf{view}_{\mathsf{MIM},2}, \mathsf{st}_2) = (\mathsf{view}_{\mathsf{MIM},3}, \{\widetilde{x}_i\}_{i \in [\widetilde{\nu}]}).$$

  Here, $\mathsf{view}_{\mathsf{MIM},2}$ *and* $\mathsf{view}_{\mathsf{MIM},3}$ *denotes the set of all messages sent in the second and third round (respectively) of the multi-party execution with* MIM*. We require that* $\{\widetilde{x}_i\}$ *(which is part of the output of* $\mathsf{Sim}^{\mathsf{MIM}}$*) is consistent*[13] *with* $(\mathsf{view}_{\mathsf{MIM},2}, \mathsf{view}_{\mathsf{MIM},3})$*.*

- *For any input* $x' = (x'_2, x'_3)$*, we require that the running time of* $\mathsf{Sim}_1^{\mathsf{MIM}}(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}, p)$ *is polynomial in* λ *and linear in* $\frac{1}{p}$*, while the running times of* $\mathsf{Sim}_2^{\mathsf{MIM}}(x'_2, \mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}_1)$ *and* $\mathsf{Sim}_3^{\mathsf{MIM}}(x'_2, x'_3, \mathsf{view}_{\mathsf{MIM},1}, \mathsf{view}_{\mathsf{MIM},2}, \mathsf{st}_2)$ *are polynomial in* λ*, independent of* p*.*

- *If* $\Pr\left[\mathsf{pExtract}_c^{\mathsf{MIM}}(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}) \geq \lambda^{-c}\right] \geq \lambda^{-c}$*, then we have:*

  $$\left(x', \mathsf{view}_{\mathsf{MIM},1}, \mathsf{IDEAL}_{\mathsf{MIM}}(x', \mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}) \middle|\ \mathsf{pExtract}_c^{\mathsf{MIM}}(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}) \geq \lambda^{-c}\right) \approx$$
  $$\left(x, \mathsf{view}_{\mathsf{MIM},1}, \mathsf{REAL}_{\mathsf{MIM}}(x, w, \mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}) \middle|\ \mathsf{pExtract}_c^{\mathsf{MIM}}(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}) \geq \lambda^{-c}\right)$$

  where $(x, w) \leftarrow (\mathcal{X}, \mathcal{W})_\lambda$*,* $x' \leftarrow \mathcal{X}'_\lambda$*, and* $(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{st})$ *is generated by simulating all the messages sent in the first round of the execution with* MIM*,*[14] *where* $\mathsf{view}_{\mathsf{MIM},1}$ *denotes all the simulated messages and* st *denotes the secret states of all the honest parties, and*

  $$\mathsf{IDEAL}_{\mathsf{MIM}}(x', \mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}) = (\mathsf{view}_{\mathsf{MIM},1}, \mathsf{view}_{\mathsf{MIM},2}, \mathsf{view}_{\mathsf{MIM},3}),$$

[13] Note that $(\mathsf{view}_{\mathsf{MIM},2}, \mathsf{view}_{\mathsf{MIM},3})$ includes the instances $\{\widetilde{x}_i\}$, and we add the instances explicitly to the output of $\mathsf{Sim}^{\mathsf{MIM}}$ only so that we will be able to refer to it later.

[14] Note that this can be simulated easily since the protocol is delayed-input which means that the parties do not use their private inputs to compute their first round message.

where the variables $(\mathsf{view}_{\mathsf{MIM},2}, \mathsf{view}_{\mathsf{MIM},3})$ *are computed by running* $\mathsf{Sim}^{\mathsf{MIM}}(x', \mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}, p)$ *for* $p = \mathsf{pExtract}_c^{\mathsf{MIM},\mathcal{D}}(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{st})$. *The experiment* $\mathsf{REAL}_{\mathsf{MIM}}(x, w, \mathsf{view}_{\mathsf{MIM},1})$ *is computed by running a real world execution with* MIM*, where the provers in the "left" sessions uses the input* (x, w) *and where the first round messages are* $\mathsf{view}_{\mathsf{MIM},1}$*, and by* $\mathsf{Valid}(\mathsf{Trans}_{\text{left}}(P_2(x, w, \mathsf{st})))$ *we mean that all left sessions in the execution of* $\mathsf{REAL}_{\mathsf{MIM}}$ *are valid.*

- *Over the randomness of* Sim*, of generating* $(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{st})$ *and over* $x' \leftarrow \mathcal{X}'_\lambda$*,*

$$\Pr\left[\bigvee_{i\in[\widetilde{\nu}]}\left(\mathsf{Acc}(\widetilde{\mathsf{Trans}}_i) = 0\right) \bigvee \left(\bigwedge_{i\in[\widetilde{\nu}]} \widetilde{x}_i \in L\right)\right] \geq 1 - \lambda^{-c},$$

where $\{\widetilde{\mathsf{Trans}}_i\}$ *is the transcript of the* i*'th right execution when* $(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{view}_{\mathsf{MIM},2}, \mathsf{view}_{\mathsf{MIM},3})$ *are computed in* $\mathsf{IDEAL}_{\mathsf{MIM}}(x', \mathsf{view}_{\mathsf{MIM},1}, \mathsf{st})$ *as above, and* $\mathsf{Acc}(\widetilde{\mathsf{Trans}}_i) = 0$ *denotes the event that the (publicly verifiable) transcript* $\widetilde{\mathsf{Trans}}_i$ *causes an honest verifier to reject.*

5.2 Constructing Simulation Sound Promise ZK

In this section, we describe our construction of Simulation Sound Promise ZK. Formally, we prove the following theorem:

Theorem 7. *Assuming the existence of polynomially secure injective one way functions, there exists a three round simulation-sound promise ZK argument according to Definition 10.*

5.2.1 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 $\mathcal{D}_\lambda = (\mathcal{X}_\lambda, \mathcal{W}_\lambda)$ be any efficiently sampleable distribution on R_L.

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

- $\mathsf{TDGen} = (\mathsf{TDGen}_1, \mathsf{TDGen}_2, \mathsf{TDGen}_3, \mathsf{TDOut})$ is the three-message trapdoor generation protocol from Section 4, that is 3-extractable according to Definition 3, with corresponding extractor TDExt.
- $\mathsf{RWI} = (\mathsf{RWI}_1, \mathsf{RWI}_2, \mathsf{RWI}_3, \mathsf{RWI}_4)$ is the three round delayed-input witness indistinguishable argument with bounded rewinding security for $L = 5$ from Definition 6. The fourth algorithm RWI_4 is the final verification algorithm.
- $\mathsf{NMCom} = (\mathsf{NMCom}_1, \mathsf{NMCom}_2, \mathsf{NMCom}_3)$ denotes a special non-malleable commitment according to Definition 4.

NP Languages. We define the following relation R' that will be useful in our construction. Parse instance $\mathsf{st} = (x, \mathsf{c}, \mathsf{td}_1)$, where $\mathsf{c} = (\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3)$. Parse witness $\mathsf{w} = (w, \mathsf{t}, \mathsf{r})$. Then, $R'(\mathsf{st}, \mathsf{w}) = 1$ if and only if :

$$\left(R(x, w) = 1\right) \bigvee \left(\mathsf{TDValid}(\mathsf{td}_1, \mathsf{t}) = 1 \wedge \mathsf{c}_1 = \mathsf{NMCom}_1(\mathsf{r}) \wedge \mathsf{c}_3 = \mathsf{NMCom}_3(\mathsf{t}, \mathsf{c}_1, \mathsf{c}_2; \mathsf{r})\right).$$ We denote the corresponding language by L'.

That is, either :

1. x is in the language L with witness w, OR,

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

We construct a three round protocol $\pi^{\mathsf{SE-PZK}} = (P, V, \mathsf{Valid})$ for L in Figure 2. The completeness of this protocol follows from the correctness of the underlying primitives.

Inputs: Prover P with tag tag obtains input $(x = (x_2, x_3), w) \leftarrow (\mathcal{X}, \mathcal{W})$ in the second round.

1. **Round 1:**
   - **Prover message:**
     - Compute $\mathsf{rwi}_1 = \mathsf{RWI}_1(1^\lambda, \widehat{r})$.
     - Sample $r \leftarrow \{0,1\}^*$.
     - Compute $\mathsf{c}_1 \leftarrow \mathsf{NMCom}_1(\mathsf{r})$ using NMCom with tag tag and uniform randomness[a].
     - Send $(\mathsf{rwi}_1, \mathsf{c}_1)$.
   - **Verifier message:**
     Sample $\mathsf{r}_{\mathsf{td}} \xleftarrow{\$} \{0,1\}^*$, then compute and send $\mathsf{td}_1 \leftarrow \mathsf{TDGen}_1(\mathsf{r}_{\mathsf{td}})$.
2. **Round 2:**
   - **Prover message:**
     - Obtain input $(x = (x_2, x_3), w)$ which is a randomly chosen sample from $(\mathcal{X}_\lambda, \mathcal{W}_\lambda)$. Send x_2 to V.
     - Compute and send $\mathsf{td}_2 \leftarrow \mathsf{TDGen}_2(\mathsf{td}_1)$.
   - **Verifier message:**
     Compute and send $\mathsf{rwi}_2 \leftarrow \mathsf{RWI}_2(\mathsf{rwi}_1)$ and $\mathsf{c}_2 \leftarrow \mathsf{NMCom}_2(\mathsf{c}_1)$.
3. **Round 3:**
   - **Prover message:**
     - Compute $\mathsf{c}_3 \leftarrow \{0,1\}^m$ and let $\mathsf{c} = (\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3)$.
     - Set $x' = (x, \mathsf{c}, \mathsf{td}_1)$ and $w' = (w, \perp, \perp)$. Compute $\mathsf{rwi}_3 \leftarrow \mathsf{RWI}_3(\mathsf{rwi}_1, \mathsf{rwi}_2, x', w')$ for $R'(x', w') = 1$ for R' defined above.
     - Send $(x_3, \mathsf{c}_3, \mathsf{rwi}_3)$.
   - **Verifier message:**
     Sample and send $\mathsf{td}_3 \leftarrow \mathsf{TDGen}_3(\mathsf{td}_1, \mathsf{td}_2, \mathsf{r}_{\mathsf{td}})$ using uniform randomness.
4. **Verifier Output:**
   Output $\mathsf{RWI}_4(\mathsf{rwi}_1, \mathsf{rwi}_2, \mathsf{rwi}_3, \mathsf{st})$.

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

[a] We omit explicit dependence of the algorithm on tag to avoid cluttering.

Figure 2: Three round Simulation-Sound Promise ZK argument.

5.3 Security Proof

We describe the simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3)$ in Figure 3. We will denote messages generated in right sessions by $\widetilde{a}$, while corresponding messages in left sessions will be denoted by a. In Figure 3 we only describe a simulator in the one-one setting, this generalizes to the many-many setting of Definition 10 by a direct hybrid argument.

The simulator $\mathsf{Sim} := (\mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3)$ generates all verifier messages for the right session according to honest verifier strategy. For the left session, Sim_1 generates messages as follows:

1. **Round 1:**
   - Sample $\mathsf{r} \xleftarrow{\$} \{0,1\}^*$, compute $\mathsf{rwi}_1 \leftarrow \mathsf{RWI}_1(1^\lambda)$, and $\mathsf{c}_1 \leftarrow \mathsf{NMCom}_1(\mathsf{r})$.
   - Send $\mathsf{msg} = (\mathsf{rwi}_1, \mathsf{c}_1)$ together with honest verifier message for the right session, and store the randomness used to generate it as st.
   - Obtain message td_1 from MIM for the left session.
2. **Look-ahead threads:**
   - Let $p = \mathsf{pExtract}_c^{\mathsf{MIM}}(\mathsf{msg}, \mathsf{st})$. If $p = 0$, output $\perp$.
   - Else, create a set of $(\lambda \cdot \frac{1}{p})$ look-ahead threads as follows, with Round 1 fixed as above.
3. **Round 2:** In thread i for $i \in [\lambda/p]$,
   - Sample independently $(x_i = (x_{2,i}, x_{3,i}), w_i) \leftarrow \mathcal{D}_\lambda$.
   - Sample independently $\mathsf{td}_{2,i} \leftarrow \mathsf{TDGen}_2(\mathsf{td}_1)$, send $(\mathsf{td}_{2,i}, x_{2,i})$.
   - Receive $\mathsf{rwi}_{2,i}$ and $\mathsf{c}_{2,i}$.
4. **Round 3:** In thread i for $i \in [\lambda/p]$,
   - Compute $\mathsf{c}_{3,i} \xleftarrow{\$} \{0,1\}^m$ and set $x'_i = (x_i, \mathsf{c}_i, \mathsf{td}_i)$, $w'_i = (\mathsf{r}_{b,i}, \perp, \perp)$, where $\mathsf{c}_i = (\mathsf{c}_1, \mathsf{c}_{2,i}, \mathsf{c}_{3,i}), \mathsf{td}_i = (\mathsf{td}_1, \mathsf{td}_{2,i}, \mathsf{td}_{3,i})$.
   - Generate $\mathsf{rwi}_{3,i} \leftarrow \mathsf{RWI}_3(\mathsf{rwi}_1, \mathsf{rwi}_2, x'_i, w'_i)$.
   - Send $(x_{3,i}, \mathsf{rwi}_{3,i})$ and receive $\mathsf{td}_{3,i}$ (which may or may not be Valid).
5. **Trapdoor Extraction:**
   - Extract trapdoor from look-aheads by computing $\mathsf{t}_V = \mathsf{TDExt}(\mathsf{td}_1, \{\mathsf{td}_{2,i}, \mathsf{td}_{3,i}\}_{i \in [\lambda/p]})$.
   - Output $\mathsf{t}_V, \mathsf{st}$ and "Special Abort" if TDExt fails.
6. For left session, $\mathsf{Sim}_2(x_2)$ samples $\mathsf{td}_2 \leftarrow \mathsf{TDGen}_2(\mathsf{td}_1)$, sends (td_2, x_2) and receives $\mathsf{rwi}_2, \mathsf{c}_2$.
7. For left session, $\mathsf{Sim}_3(x_3, \mathsf{t}_V, \mathsf{st})$ does the following: Compute $\mathsf{c}_3 \leftarrow \mathsf{NMCom}_3(\mathsf{c}_1, \mathsf{c}_2, \mathsf{t}_V)$ and $\mathsf{rwi}_3 \leftarrow \mathsf{RWI}_3(\mathsf{rwi}_1, \mathsf{rwi}_2, x', w')$, where $x' = (x, \mathsf{c}, \mathsf{td}), w' = (\perp, \mathsf{t}_V, \mathsf{r})$, $\mathsf{c} = (\mathsf{c}_1, \mathsf{c}_2, \mathsf{c}_3), \mathsf{td} = (\mathsf{td}_1, \mathsf{td}_2, \mathsf{td}_3)$. Send $(x_3, \mathsf{c}_3, \mathsf{rwi}_3)$ and output the view of the MIM.

Figure 3: Simulator algorithm Sim.

5.3.1 Analysis of the Simulator: Hybrids

Suppose that there exists a constant $c > 0$ and a distinguisher that distinguishes between the real and ideal games (with parameter c) with noticeable probability ϵ. We will now prove indistinguishability of real and ideal games by using a sequence of hybrids. We will also maintain an invariant that will help us prove the (main) simulation-extraction property.

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_4 corresponds to the ideal world where MIM interacts with the simulator Sim.

- Hyb_0 **- Real World:** In this hybrid, the first round messages are all honestly executed with the adversary MIM. Then $\mathsf{pExtract}_c$ is called on this view, and it produces an output p. If $p = 0$, the experiment aborts and outputs $\perp$. If not, the challenger plays the role of the honest prover on the left session and honest verifier on the right session.

- Hyb_1 **- Trapdoor Extraction:** In this hybrid, if the value $p > 0$, then the challenger $\mathcal{C}$ creates a fresh set of $(\lambda \cdot \frac{1}{p})$ look-ahead threads on the left session. In all the look-ahead threads, $\mathcal{C}$ performs each thread exactly as described in Hyb_0 using fresh randomness. Additionally, using the messages in the look-ahead threads, $\mathcal{C}$ also runs the "Trapdoor Extraction" phase described in Step 5 of the description of Sim to extract the trapdoor t_V. It outputs "Special Abort" if trapdoor extraction fails. Finally, $\mathcal{C}$ completes the main thread on the left session by running the honest prover strategy exactly as in Hyb_0.

  Remark 3. *Looking ahead, we will maintain the invariant that the* MIM *will not be able to commit to the trapdoor of the honest verifier in the right session, except with negligible probability. This will follow by various arguments, leveraging non-malleability of the* NMCom *commitment (for* Hyb_2 *below), bounded-rewinding security of the WI arguments (for* Hyb_3 *below), and an extractability argument for the* NMCom *commitment (for* Hyb_4*) below.*

- Hyb_2 **- Changing Commitment to Trapdoor in Main Thread:** In the main thread, on the left sessions, $\mathcal{C}$ generates the NMCom commitment message $\mathsf{c}_{t,3}$ to commit to the extracted trapdoor t_V.

- Hyb_3 **- Switching bounded-rewinding secure WI arguments in Main Thread:** In the main thread, on the left sessions, $\mathcal{C}$ uses the witness t_V committed in c_t in the bounded-rewinding secure WI arguments.

- Hyb_4 **- Switching instances in Main Thread:** In the main thread, on the left sessions, $\mathsf{Sim}_{\mathsf{Hyb}}$ on the main thread, obtains input an instance x' sampled from the distribution $\mathcal{X}'_\lambda$. The description of $\mathcal{C}$ in this hybrid matches the description of the simulator Sim.

5.3.2 Invariant

We now describe the invariant.

Definition 11 (Invariant T)**.** *Consider a right session between* MIM *and verifier* V*. Here,* td_1 *denotes the first message of the trapdoor generation protocol with* V *as the trapdoor generator. The values* $(\widetilde{\mathsf{c}}_{t,1}, \widetilde{\mathsf{c}}_{t,2}, \widetilde{\mathsf{c}}_{t,3})$ *denote the messages of the non-malleable commitment generated by the* MIM *in the right session. We say that the event* E_t *occurs if* $\exists(\widetilde{\mathsf{t}}, \widetilde{\mathsf{r}}_t, \widetilde{r})$ *such that:*

$$\big(\widetilde{\mathsf{c}}_{t,1} = \mathsf{NMCom}_1(\widetilde{\mathsf{r}}_t; \widetilde{r})\big) \wedge \big(\widetilde{\mathsf{c}}_{t,3} = \mathsf{NMCom}_3(\widetilde{\mathsf{t}}; \widetilde{\mathsf{r}}_t)\big) \wedge \big(\mathsf{TDValid}(\widetilde{\mathsf{td}}_1, \widetilde{\mathsf{t}}) = 1\big).$$

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* $\widetilde{\mathsf{t}}$ *for the trapdoor generation messages of* V*.*

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

5.3.3 Hybrid Security Argument

In this section, we will establish simulation soundness by analyzing the hybrids presented above. First, recall that we suppose that there exists a constant $c > 0$ for which an adversary $\mathcal{A}$ distinguishes between the real and ideal games (with parameter c) with noticeable probability. Moreover, since all hybrids are identical when $\mathsf{pExtract}_c = 0$, note that $\mathcal{A}$ must distinguish conditioned on $\mathsf{pExtract}_c^{\mathsf{MIM}}(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}) \geq \lambda^{-c}$.

We now prove that every pair of consecutive hybrids conditioned on $\mathsf{pExtract}_c^{\mathsf{MIM}}(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}) \geq \lambda^{-c}$ is indistinguishable, and this completes the proof.

Claim 1. *Assuming "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 does not hold in Hyb_0. That is, there exists an MIM such that it causes event E_t to occur with non-negligible probability $q(\lambda) = \frac{1}{\mathsf{poly}(\lambda)}$ for some polynomial $\mathsf{poly}(\cdot)$. 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. Before describing $\mathcal{A}_{\mathsf{TDGen}}$, we recall from Imported Theorem 1 that the NMCom satisfies the 2-extractability property from Definition 3, and we denote the extraction algorithm that on input two transcripts outputs the extracted value with probability $\frac{1}{p(\lambda)}$ for some polynomial $p(\cdot)$, by NMExt. The adversary $\mathcal{A}_{\mathsf{TDGen}}$ behaves as follows.

- Obtain input first round message t_1 corresponding to the protocol TDGen.
- Interact with the MIM emulating the role of challenger $\mathcal{C}$ in Hyb_0. Set the verifier first message for right interaction to $\widetilde{\mathsf{td}}_1$, generate all other messages according to Hyb_0 for rounds 1 and 2.
- On obtaining the MIM's second round message, parse it to obtain value $\widetilde{\mathsf{td}}_2$. Forward $\widetilde{\mathsf{td}}_2$ as the second message of TDGen.
- On input t_3, set the third TDGen message of the verifier for the right session $\widetilde{\mathsf{td}}_3 = t_3$. Generate all other messages for this round according to Hyb_0, and record the non-malleable commitment $\widetilde{\mathsf{c}}_{t,1}, \widetilde{\mathsf{c}}_{t,2}, \widetilde{\mathsf{c}}_{t,3}$ of the MIM.
- Rewind MIM back to the end of round 1 (note that no rewinding occurs in the actual Hyb_0, we only rewind for the purpose of the reduction).
- Generate fresh messages for round 2 according to the strategy in Hyb_0.
- As in the main thread, on obtaining the MIM's second round message, parse it to obtain value $\widetilde{\mathsf{td}}'_2$. Forward $\widetilde{\mathsf{td}}'_2$ as the (rewinding) second message of TDGen.
- On input t'_3, set the third trapdoor generation message of the verifier for the right session $\widetilde{\mathsf{td}}'_3 = t'_3$. Generate all other messages for this round according to Hyb_0, and record the non-malleable commitment $\widetilde{\mathsf{c}}_{t,1}, \widetilde{\mathsf{c}}'_{t,2}, \widetilde{\mathsf{c}}'_{t,3}$ of the MIM.

- Run $\mathsf{NMExtract}(\widetilde{\mathsf{c}}_{t,1}, \widetilde{\mathsf{c}}_{t,2}, \widetilde{\mathsf{c}}_{t,3}, \widetilde{\mathsf{c}}'_{t,2}, \widetilde{\mathsf{c}}'_{t,3})$ and output the message m extracted from both (which could be $\perp$).

By the 2-extractability property, there exists a polynomial $\widetilde{p}(\cdot)$, such that in this experiment, if the MIM committed to the trapdoor in t_1 with probability q, $\mathcal{A}_{\mathsf{TDGen}}$ outputs t_1 with probability at least $\widetilde{p} \cdot q$. On the other hand, the 1-rewinding security of TDGen implies that no adversary has advantage greater than $\mathsf{negl}(\lambda)$ in guessing t_1: this gives a contradiction and completes the proof of the claim. □

Claim 2. *Assuming Claim 1, invariant T holds in Hyb_1.*

Proof. Since the main threads in Hyb_0 and Hyb_1 are identically distributed, the invariant continues to hold in Hyb_1. □

Claim 3. $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$.

Proof. The only difference between Hyb_0 and Hyb_1 is statistical: the challenger outputs an additional "Special Abort" in Hyb_1 if trapdoor extraction fails in Step 5. By Definition 10, the adversary is promised to output valid transcripts in the left session with probability at least $\frac{p}{2}$. Thus, given $\frac{\lambda}{p}$ rewinding executions, the adversary outputs at least 2 valid transcripts with probability $1-(1-\frac{p}{2})^{\frac{\lambda}{p}} \geq 1-\exp^{-\lambda}$. This means that Hyb_0 and Hyb_1 are at most $\mathsf{negl}(\lambda)$-apart, proving the claim. □

Claim 4. *Assuming NMCom is a special non-malleable commitment scheme according to Definition 4, invariant T holds in Hyb_2.*

Proof. The only difference between Hyb_1 and Hyb_2 is that in Hyb_2, the simulator computes the non-malleable commitment in the main thread of the left session using the adversary's trapdoor. Assume for the sake of contradiction that there exists a MIM that causes event E_t to occur with probability $q = \frac{1}{\mathsf{poly}(\cdot)}$ in Hyb_2 for some polynomial $\mathsf{poly}(\cdot)$. We will use MIM to design an adversary $\mathcal{A}_{\mathsf{NMCom}}$ that breaks the security of the non-malleable commitment scheme. $\mathcal{A}_{\mathsf{NMCom}}$ performs the role of $\mathcal{C}$ in its interaction with MIM and behaves as follows.

- Obtain input the first round message of the left NMCom, and set $\mathsf{c}_{t,1}$ to this message. Generate all other messages for the first round same as Hyb_1, using honest prover and verifier strategy.
- Start creating lookahead threads exactly as in Hyb_1. Note that in each lookahead thread, the third message of NMCom for the left session is sampled uniformly at random and independent of the first two messages.
- Extract trapdoor t_V according to Step 5 from the look-ahead threads.
- Run the second round of the main thread exactly as in Hyb_1, and output the MIM's second round challenge $\mathsf{c}_{t,2}$.
- Sample r uniformly at random and output (t_V, r) to the NMCom challenger.
- On input the third message $\mathsf{c}_{t,3}$ of NMCom, generate the third round of the protocol using $\mathsf{c}_{t,3}$ in the left non-malleable commitment and generating other messages according to Hyb_1.

By the non-malleability of NMCom, the joint distribution of the view and value committed by the MIM remains indistinguishable between Hyb_1 and Hyb_2. Thus, invariant T holds in Hyb_2. □

Claim 5. $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$.

Proof. The main difference between Hyb_1 and Hyb_2 is that in Hyb_2, the simulator computes the non-malleable commitment using the adversary's trapdoor value in the left session. Suppose there exists an adversary V^* that can distinguish the two hybrids with noticeable probability, we can use V^* to design an adversary $\mathcal{A}_{\mathsf{Hid}}$ (identical to $\mathcal{A}_{\mathsf{NMCom}}$ in Claim 4 above) that breaks the hiding of the non-malleable commitment scheme. □

Claim 6. *Assuming* RWI *is bounded rewinding witness indistinguishable according to Definition 6,* TDGen *is* 3*-extractable according to Definition 3, and the special non-malleable commitment* NMCom *is* 2*-extractable according to Definition 4, invariant T holds in* Hyb_3.

Proof. The only difference between Hyb_2 and Hyb_3 is that in Hyb_3, in the main thread of the left session, the simulator computes the WI proof using the trapdoor witness. Assume for the sake of contradiction that there exists an MIM adversary that causes event E_t to occur with probability q in Hyb_3, where $q = \frac{1}{\mathsf{poly}(\cdot)}$ for some polynomial $\mathsf{poly}(\cdot)$. 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 the MIM as challenger, behaving as follows:

- Obtain input the first message rwi_1 of the bounded rewinding WI argument. Send this as the first round message on behalf of honest prover, and generate all other first round messages according to Hyb_2.
- Instead of executing λ/p lookahead threads, execute exactly 2 look-ahead threads. For both threads, obtain RWI messages externally using witness $(w, \bot, \bot)$.
- Set all the values $\{\mathsf{c}_{t,2,j}, \mathsf{c}_{t,3,j}\}_{j\in[3,\lambda/p]}$ to $\bot$, and run $\mathsf{TDGen}(\mathsf{c}_{t,1}, \{\mathsf{c}_{t,2,j}, \mathsf{c}_{t,3,j}\}_{j\in[\lambda/p])}$ to output t_V (which may be $\bot$).
- If $\mathsf{t}_V = \bot$, abort, else generate rounds 2 and 3 for the main thread according to Hyb_2 using t_V. Obtain the RWI message for use in the left execution, externally, to use witness $(w, \bot, \bot)$ or $(\bot, \mathsf{t}_V, r_t)$ depending on the verifier challenge. Record the MIM's non-malleable commitment $(\widetilde{\mathsf{c}}_{t,1}, \widetilde{\mathsf{c}}_{t,2}, \widetilde{\mathsf{c}}_{t,3})$.
- Next, rewind to the end of round 1 and again generate rounds 2 and 3 (with fresh uniform randomness) according to Hyb_2 using t_V. Obtain the RWI message for use in the left execution, externally, to use witness $(w, \bot, \bot)$ or $(\bot, \mathsf{t}_V, r_t)$ depending on the verifier challenge. Record the MIM's non-malleable commitment $(\widetilde{\mathsf{c}}_{t,1}, \widetilde{\mathsf{c}}'_{t,2}, \widetilde{\mathsf{c}}'_{t,3})$.
- Output $\mathsf{NMExtract}(\widetilde{\mathsf{c}}_{t,1}, \widetilde{\mathsf{c}}_{t,2}, \widetilde{\mathsf{c}}_{t,3}, \widetilde{\mathsf{c}}'_{t,2}, \widetilde{\mathsf{c}}'_{t,3})$ (which may be $\bot$).

By assumption on Hyb_3, by 2-extractability of NMCom and 3-extractability of the extractable commitment, when RWI uses witness $(\bot, \mathsf{t}_V, r_t)$ in the main threads, we have that $\mathcal{A}_{\mathsf{RWI}}$ outputs the honest party trapdoor t with probability $\frac{q}{\mathsf{poly}(\lambda)}$ for some polynomial $\mathsf{poly}(\cdot)$. On the other hand, by Claim 4 for Hyb_2, $\mathcal{A}_{\mathsf{RWI}}$ outputs the honest party trapdoor t with probability $\mathsf{negl}(\lambda)$ when RWI uses witness $(w, \bot, \bot)$ in the main threads. This contradicts the bounded rewinding security of WI according to Definition 6. □

Claim 7. $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$.

Proof. The main difference between Hyb_2 and Hyb_3 is that in Hyb_2, the simulator computes the bounded rewinding WI argument using the adversary's trapdoor value in the left session. Suppose there exists an adversary V^* that can distinguish the two hybrids with noticeable probability, we can use V^* to design an adversary $\mathcal{A}_{\mathsf{WI}}$ (identical to $\mathcal{A}_{\mathsf{RWI}}$ in Claim 6 above except that it does not perform the last step) that breaks the bounded rewinding security of the WI. □

Claim 8. *Assuming* 2*-extractability of the* NMCom *according to Definition 4, invariant* T *holds in* Hyb_4.

Proof. The only difference between Hyb_3 and Hyb_4 is that in Hyb_4, in the main thread of the left session, the simulator uses instance $x' \xleftarrow{\$} \mathcal{X}'$ instead of $x \xleftarrow{\$} \mathcal{X}$. Assume for the sake of contradiction that there exists an MIM adversary that causes event E_t to occur with probability q in Hyb_4, where $q = \frac{1}{\mathsf{poly}(\cdot)}$ for some polynomial $\mathsf{poly}(\cdot)$. We will use $\mathcal{A}$ to design an adversary $\mathcal{A}_x$ that breaks the indistinguishability between distributions $\mathcal{X}$ and $\mathcal{X}'$.

$\mathcal{A}_{\mathsf{RWI}}$ interacts with the MIM as challenger, behaving as follows:

- Obtain input x (that is sampled from one out of $\mathcal{X}$ or $\mathcal{X}'$).
- Generate lookahead threads according to Hyb_3, and obtain trapdoor t_V.
- Just as in Hyb_3, if $\mathsf{t}_V = \bot$, abort, else generate rounds 2 and 3 for the main thread according to Hyb_3 using t_V for instance x.
- Next, rewind to the end of round 1 and again generate rounds 2 and 3 (with fresh uniform randomness) according to Hyb_2 using t_V, and using instance x. Record the MIM's non-malleable commitment $(\widetilde{\mathsf{c}}_{t,1}, \widetilde{\mathsf{c}}'_{t,2}, \widetilde{\mathsf{c}}'_{t,3})$.
- Output $\mathsf{NMExtract}(\widetilde{\mathsf{c}}_{t,1}, \widetilde{\mathsf{c}}_{t,2}, \widetilde{\mathsf{c}}_{t,3}, \widetilde{\mathsf{c}}'_{t,2}, \widetilde{\mathsf{c}}'_{t,3})$ (which may be $\bot$).

By assumption on Hyb_4, by 2-extractability of NMCom, when RWI uses witness $(\bot, \mathsf{t}_V, r_t)$ in the main threads, we have that $\mathcal{A}_x$ outputs the honest party trapdoor t with probability $\frac{q}{\mathsf{poly}(\lambda)}$ for some polynomial $\mathsf{poly}(\cdot)$. On the other hand, by Claim 6 for Hyb_3, $\mathcal{A}_x$ outputs the honest party trapdoor t with probability $\mathsf{negl}(\lambda)$. This contradicts the indistinguishability of distributions $\mathcal{X}$ and $\mathcal{X}'$. □

Claim 9. $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$.

Proof. The only difference between these hybrids is whether the simulator obtains $x \leftarrow \mathcal{X}$ or $x' \leftarrow \mathcal{X}'$. By indistinguishability of the distributions $\mathcal{X}$ and $\mathcal{X}'$, the two hybrids are indistinguishable. □

This proves that the real and ideal distributions remain indistinguishable. Moreover, by Claim 8 and by soundness of WI, it holds that the MIM can only output accepting proofs for $x \in L$ in the right session. This completes the proof of simulation soundness.

6 Three Round List Coin Tossing

We first define the notion of list coin tossing.

Definition 12. *An* n*-party protocol* π *in the simultaneous message setting is a* secure list coin tossing protocol *if for every PPT adversary* $\mathcal{A}$ *corrupting at most* $n-1$ *parties, there exists an expected PPT simulator* $\mathcal{S}$ *such that the output of the experiments* $\mathsf{REAL}_{\mathcal{A}}$ *and* $\mathsf{IDEAL}_{\mathcal{S}}$*, defined*

below, are computationally indistinguishable. In the real world, we denote by $(\mathsf{c}, \mathsf{view}_{\mathcal{A}})$ *the pair, where* $\mathsf{view}_{\mathcal{A}}$ *is the view of the adversary* $\mathcal{A}$ *interacting with honest parties in the protocol* π*, and* $\mathsf{c} \in \{0,1\}^{\ell} \cup \{\perp\}$ *is the output of the honest parties in this protocol execution. Similarly, we use the pair* $(\tilde{\mathsf{c}}, \mathsf{view}_{\mathcal{S}})$ *in the ideal world, defined in Figure 1 below.*

We use ℓ *to denote the length of each (honest) party's output after the running the protocol (assuming the parity did not output the abort symbol* $\perp$*).*

$\mathsf{REAL}_{\mathcal{A}}(1^{\lambda}, 1^{\ell})$	$\mathsf{IDEAL}_{\mathcal{S}}(1^{\lambda}, 1^{\ell})$
$(\mathsf{c}, \mathsf{view}_{\mathcal{A}}) \leftarrow \mathsf{REAL}_{\mathcal{A}}(1^{\lambda}, 1^{\ell})$	$1^{k} \leftarrow \mathcal{S}(1^{\lambda}, 1^{\ell})$
	$(\mathsf{c}_1, \ldots, \mathsf{c}_k) \leftarrow \{0,1\}^{\ell \cdot k}$
	$(\tilde{\mathsf{c}}, \mathsf{view}_{\mathcal{S}}) \leftarrow \mathcal{S}(\mathsf{c}_1, \ldots, \mathsf{c}_k, 1^{\lambda}, 1^{\ell})$
Output $(\mathsf{c}, \mathsf{view}_{\mathcal{A}})$	If $\tilde{\mathsf{c}} \in \{\mathsf{c}_1, \ldots, \mathsf{c}_k, \perp\}$, then output $(\tilde{\mathsf{c}}, \mathsf{view}_{\mathcal{S}})$
	Else, output fail.

Table 1: List Coin Tossing

Definition 13. *We say that an* n*-party protocol* π *is a secure list coin tossing protocol in the simultaneous message setting with* black-box simulation *if there exists a (universal) expected PPT oracle machine* $\mathcal{S}$ *such that for every PPT adversary* $\mathcal{A}$ *corrupting at most* $n-1$ *parties, the simulator* $\mathcal{S}^{\mathcal{A}}$ *satisfies that*

$$\mathsf{REAL}_{\mathcal{A}}(1^{\lambda}, 1^{\ell}) \approx \mathsf{IDEAL}_{\mathcal{S}^{\mathcal{A}}}(1^{\lambda}, 1^{\ell})$$

where $\mathsf{REAL}_{\mathcal{A}}(1^{\lambda}, 1^{\ell})$ *and* $\mathsf{IDEAL}_{\mathcal{S}^{\mathcal{A}}}(1^{\lambda}, 1^{\ell})$ *are defined as in Definition 12.*

Theorem 8. *There exists a 3-round secure multi-party list coin tossing protocol* π *in the simultaneous message setting with* black-box simulation*, assuming the existence of a simulation extractable promise ZK scheme for NP.*

This, together with Theorem 7, implies the following corollary.

Corollary 9. *There exists a 3-round secure multi-party list coin tossing protocol* π *in the simultaneous message setting with* black-box simulation*, assuming the existence of injective one-way functions.*

As mentioned earlier, by applying the transformation of [GMPP16][15] to the protocol from Corollary 9 for the 2-party case, we can obtain a 4-round 2-party list coin-tossing protocol in the *unidirectional-message* model. This result overcomes the barrier of five rounds for standard 2-party coin-tossing established by [KO04].

Corollary 10 (Informal)**.** *Assuming the existence of injective one-way functions, there exists a 4-round 2-party secure list coin-tossing protocol in the unidirectional-message model (with black-box simulation).*

The rest of this section is devoted to proving Theorem 8. We provide the construction of our list coin-tossing protocol in Section 6.1, and provide the security proof in Section 6.2.

[15]The work of Garg et al. [GMPP16] establishes an impossibility result for 3-round multi-party coin-tossing by transforming any 3-round two-party coin-tossing protocol in the simultaneous-message model into a 4-round two-party coin-tossing protocol in the unidirectional-message model, and then invoking the impossibility of [KO04].

6.1 Our List Coin Tossing Protocol

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 any simulation extractable promise ZK argument system (see Section 5).

We first list some notation and building blocks that we use use in our protocol.

Building Blocks

- Let Com be any non-interactive commitment scheme. We know that such a commitment scheme can be built assuming injective one way functions.
- $\pi^{\mathsf{SE-PZK}} = (P_{\mathsf{ZK}}, V_{\mathsf{ZK}}, \mathsf{Valid}, \mathcal{E})$ any 3-round simulation-extractable delayed-input distributional promise ZK argument system (such as the one constructed in Section 5).

  For all $i \in [3]$, we use $P_{\mathsf{ZK}i}$ to denote the algorithm used by the prover P to compute the i^{th} round messages, and we use $V_{\mathsf{ZK}i}$ to denote the algorithm used by the verifier V to compute the i^{th} round messages. Further, let $V_{\mathsf{ZK}4}$ 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 the NP language L characterized by the following relation R.
Statement : $\mathsf{x} = (\mathsf{c}, \mathsf{r})$
Witness : $\mathsf{w} = \mathsf{s}$
$R(\mathsf{x}, \mathsf{w}) = 1$ 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 to party P_j, where party P_i is taking the role of the prover in the underlying promise-ZK protocol and P_j is taking the role of the verifier. We use $j \leftarrow i$ to denote that the message was sent by party P_i to party P_j, where party P_i is taking the role of the verifier in the underlying promise-ZK protocol and P_j is taking the role of the prover. (Recall that all messages are broadcast).
- The round number of the sub-protocol $\pi^{\mathsf{SE-PZK}}$ being used is written in the subscript.

Our 3-round list coin tossing protocol π^{CT} is described in Figure 4.

6.2 Security Proof

In this section, we prove Theorem 8.

Proof. Fix any adversary $\mathcal{A}$ who corrupts at most $n-1$ parties in the protocol π^{CT}. Assume without loss of generality that $\mathcal{A}$ corrupts exactly $n-1$ parties. This is without loss of generality since in the case of input-less functionalities allowing the adversary to corrupt more parties strictly makes the result stronger. Assume without loss of generality that party P_1 is the only honest party. Moreover, assume without loss of generality that $\mathcal{A}$ is deterministic. This is without loss of generality since we can think of the coins of $\mathcal{A}$ as being fixed and hardwired. Let $\mathcal{E}$ denote the

1. **Round 1:** $\forall j \in [n]$ with $j \neq i$, P_i does the following:

   - Choose random strings $r^{i\to j}, r^{j\leftarrow i} \leftarrow \{0,1\}^\lambda$ and compute $\mathsf{prove}_1^{i\to j} = P_{\mathsf{ZK}1}(1^\lambda, r^{i\to j})$ and $\mathsf{ver}_1^{j\leftarrow i} \leftarrow V_{\mathsf{ZK}1}(1^\lambda, r^{j\leftarrow i})$.
   - Broadcast $\left\{\mathsf{prove}_1^{i\to j}, \mathsf{ver}_1^{j\leftarrow i}\right\}_{j\neq i}$.

   For every $i \neq j$, let $\mathsf{view}_1^{i\rightleftarrows j} \triangleq (\mathsf{prove}_1^{i\to j}, \mathsf{ver}_1^{i\leftarrow j})$.

2. **Round 2:** P_i does the following for each $j \in [n]$ with $j \neq i$:

   - Compute $\mathsf{prove}_2^{i\to j} = P_{\mathsf{ZK}2}(\mathsf{view}_1^{i\rightleftarrows j}; r^{i\to j})$ and $\mathsf{ver}_2^{j\leftarrow i} = V_{\mathsf{ZK}2}(\mathsf{view}_1^{i\rightleftarrows j}; r^{j\leftarrow i})$.
   - Choose a random string $\mathsf{r}_i \leftarrow \{0,1\}^\ell$ and a random string $\mathsf{s}_i \leftarrow \{0,1\}^\lambda$, and compute $\mathsf{c}_i = \mathsf{Com}(\mathsf{r}_i; \mathsf{s}_i)$.
   - Broadcast $\left(\mathsf{c}_i, \left\{\mathsf{prove}_2^{i\to j}, \mathsf{ver}_2^{j\leftarrow i}\right\}_{j\neq i}\right)$.

   For every $i \neq j$, let $\mathsf{view}_2^{i\rightleftarrows j} \triangleq (\mathsf{prove}_2^{i\to j}, \mathsf{ver}_2^{i\leftarrow j})$.

3. **Round 3:** P_i does the following:

   - For each $j \in [n]$ with $j \neq i$, do:
     - Generate $\mathsf{prove}_3^{i\to j} = P_{\mathsf{ZK}3}(\mathsf{view}_1^{i\rightleftarrows j}, \mathsf{view}_2^{i\rightleftarrows j}, (\mathsf{c}_i, \mathsf{r}_i, \mathsf{s}_i); r^{i\to j})$, for the statement $(c_i, r_i) \in L$ using the witness s_i.
     - Generate $\mathsf{ver}_3^{j\leftarrow i} \leftarrow V_{\mathsf{ZK}3}(\mathsf{view}_1^{i\rightleftarrows j}, \mathsf{view}_2^{i\rightleftarrows j}; r^{j\leftarrow i})$.
   - Broadcast $\left(\mathsf{r}_i, \left\{\mathsf{prove}_3^{i\to j}, \mathsf{ver}_3^{j\leftarrow i}\right\}_{j\neq i}\right)$.

   For every $i \neq j$, let $\mathsf{view}_3^{i\rightleftarrows j} \triangleq (\mathsf{prove}_3^{i\to j}, \mathsf{ver}_3^{i\leftarrow j})$.

4. **Output Computation:** Each party checks if there exists $i \neq j$ such that

$$\mathsf{Valid}\left(\mathsf{view}_1^{i\rightleftarrows j}, \mathsf{view}_2^{i\rightleftarrows j}, \mathsf{view}_3^{i\rightleftarrows j}, (\mathsf{c}_i, \mathsf{r}_i)\right) = 0.$$

   If so, output $\perp$, and otherwise, output $\mathsf{r} = \bigoplus_{i\in[n]} \mathsf{r}_i$.

Figure 4: 3-round secure protocol π^{CT} for list coin tossing.

simulator (and extractor) of the simulation extractable promise ZK system as defined in Section 5 (see Definition 10). We construct a simulator $\mathcal{S}$ for the ideal world, that has oracle access to $\mathcal{A}$, as follows:

Description of the simulator $\mathcal{S}^{\mathcal{A}}(1^\lambda, 1^\ell)$.

1. **First Round Simulation:** Simulate all the messages sent in the first round, by simulating the messages of honest party honestly, and using the oracle access to $\mathcal{A}$ to simulate the messages sent by the corrupted parties. Denote all these messages by $\mathsf{view}_{\mathcal{A},1}$, and denote

the secret state of the honest party by st. (This state will be needed to continue with the simulation.)

2. **Check if Abort:** Continue to run the protocol π^{CT} to completion, while continuing to simulate the honest party using the secret state st, and using oracle access to $\mathcal{A}$ to simulate the messages sent by corrupted parties. If at the end of this execution the honest parties output $\perp$, then $\mathcal{S}$ outputs $(\perp, \mathsf{view}_{\mathcal{A}})$ where $\mathsf{view}_{\mathcal{A}}$ consists of all the messages exchanged in the simulated protocol. Otherwise, continue.

3. **Validity Approximation:** Estimate the probability that all the zero-knowledge proofs are valid, conditioned on the first round messages being $\mathsf{view}_{\mathcal{A},1}$. This is done by creating a set of look-ahead threads that run only the second and third rounds of π^{CT}, until receiving non-aborting transcripts in λ threads. Formally, this is done as follows.

   (a) Set counters $i = 0$ and $T = 0$.

   (b) Set $T = T + 1$.

   (c) Use st to simulate the protocol π^{CT} to completion, while conditioning on the messages of the first round being $\mathsf{view}_{\mathcal{A},1}$.

   (d) If the honest parties output $\perp$ in this execution, then set $i = i + 1$.

   (e) If $i = \lambda$ then output $p \triangleq \frac{\lambda}{T}$. Otherwise, go back to Item (b).

4. **Learning the Adversary's Randomness:** Denote by $\mathsf{view}^{\text{partial}}_{\mathcal{A},1}$ only the messages in $\mathsf{view}_{\mathcal{A},1}$, sent between parties P_1 and P_j (recall that we assume that P_1 is the only honest party). $\mathcal{S}^{\mathcal{A}}(1^{\lambda}, 1^{\ell}, \mathsf{view}_{\mathcal{A},1})$ does the following:

   Generate a valid look ahead thread using the extractor $\mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3)$ from Definition 10, as follows.

   (a) Run $\mathcal{E}_1^{\mathcal{A}}(\mathsf{view}^{\text{partial}}_{\mathcal{A},1}, \mathsf{st}, p) = \widetilde{\mathsf{st}}_1$.

   (b) Choose at random $\mathsf{r}_1 \leftarrow \{0,1\}^{\ell}$ and $\mathsf{s} \leftarrow \{0,1\}^{\lambda}$, and let $\mathsf{c}_1 = \mathsf{Com}(\mathsf{r}_1; \mathsf{s})$.

   (c) Run $\mathcal{E}_2^{\mathcal{A}}(\mathsf{c}_1, \mathsf{view}^{\text{partial}}_{\mathcal{A},1}, \widetilde{\mathsf{st}}_1) = (\mathsf{view}^{\text{partial}}_{\mathcal{A},2}, \widetilde{\mathsf{st}}_2)$, where the oracle is w.r.t. $\mathcal{A} = \mathcal{A}(\mathsf{view}_{\mathcal{A},1})$.

   (d) Let $\mathsf{view}_{\mathcal{A},2}$ denote the unique extension of $\mathsf{view}^{\text{partial}}_{\mathcal{A},2}$ that is consistent with $(\mathsf{view}_{\mathcal{A},1}, \mathsf{view}^{\text{partial}}_{\mathcal{A},2})$.

   Note that the only messages in $\mathsf{view}_{\mathcal{A},2}$ that are not in $\mathsf{view}^{\text{partial}}_{\mathcal{A},2}$ are messages sent between malicious players, and since we assume (without loss of generality) that $\mathcal{A}$ is deterministic, $\mathsf{view}_{\mathcal{A},2}$ is a deterministic function of $(\mathsf{view}_{\mathcal{A},1}, \mathsf{view}^{\text{partial}}_{\mathcal{A},2})$.

   (e) Run $\mathcal{E}_3^{\mathcal{A}}(\mathsf{r}_1, \mathsf{view}^{\text{partial}}_{\mathcal{A},1}, \mathsf{view}^{\text{partial}}_{\mathcal{A},2}, \widetilde{\mathsf{st}}_2)$, where the oracle is w.r.t. $\mathcal{A} = \mathcal{A}(\mathsf{view}_{\mathcal{A},1}, \mathsf{view}_{\mathcal{A},2})$.

   (f) Let $\mathsf{view}_{\mathcal{A},3}$ denote the unique extension of $\mathsf{view}^{\text{partial}}_{\mathcal{A},3}$ that is consistent with $(\mathsf{view}_{\mathcal{A},1}, \mathsf{view}_{\mathcal{A},2}, \mathsf{view}^{\text{partial}}_{\mathcal{A},3})$.

   (g) If $\mathsf{Valid}(\mathsf{view}_{\mathcal{A},1}, \mathsf{view}_{\mathcal{A},2}, \mathsf{view}_{\mathcal{A},3}) = 0$, then goto Step (4b). Otherwise, $\mathsf{view}_{\mathcal{A},2}$ includes commitments $\{\mathsf{c}_j\}_{j\in[n]}$, and $\mathsf{view}_{\mathcal{A},3}$ includes the (supposedly) committed values $\{\mathsf{r}_j\}_{j\in[n]}$.

   (h) Output $\{\mathsf{r}_j\}_{j\in[n]}$.

5. **Second Validity Approximation:** Estimate the probability that the continuation of $(\mathsf{view}_{\mathcal{A},1}, \mathsf{view}_{\mathcal{A},2})$ by $\mathcal{E}_3$ is valid. This is done by creating a set of look-ahead threads that run only the third round of π^{CT}, until it receives non-aborting transcripts in λ threads. Formally,

(a) Set counters $i = 0$ and $T = 0$.

(b) Set $T = T + 1$.

(c) Run $\mathcal{E}_3^{\mathcal{A}}(\mathsf{r}_1, \mathsf{view}_{\mathcal{A},1}^{\text{partial}}, \mathsf{view}_{\mathcal{A},2}^{\text{partial}}, \widetilde{\mathsf{st}}_2) = \mathsf{view}_{\mathcal{A},3}$, where the oracle is w.r.t. $\mathcal{A} = \mathcal{A}(\mathsf{view}_{\mathcal{A},1}, \mathsf{view}_{\mathcal{A},2})$.

(d) $\mathsf{Valid}(\mathsf{view}_{\mathcal{A},1}, \mathsf{view}_{\mathcal{A},2}, \mathsf{view}_{\mathcal{A},3}) = 1$ then set $i = i + 1$.

(e) If $i = \lambda$ then output $q \triangleq \frac{\lambda}{T}$. Otherwise, go back to Item (b).

6. **Query to Ideal Functionality:** Set $k = \lceil \frac{\lambda}{q} \rceil$, and send 1^k to the ideal functionality, and receive k random strings $\{\mathsf{ans}_1, \ldots, \mathsf{ans}_k\}$, each of length ℓ.

7. **Forcing the Output:** $\mathcal{S}^{\mathcal{A}}(1^\lambda, 1^\ell, \mathsf{view}_{\mathcal{A},1}, \mathsf{view}_{\mathcal{A},2}, \widetilde{\mathsf{st}})$ does the following:

   (a) Set counter $i = 1$.

   (b) If $i > k$ then output $\perp$ and abort.

   (c) Otherwise, set $\mathsf{r}_1^* = \mathsf{ans}_i \bigoplus \left(\bigoplus_{j=2}^{n} \mathsf{r}_j \right)$.

   (d) Run $\mathcal{E}_3^{\mathcal{A}}(\mathsf{r}_1^*, \mathsf{view}_{\mathcal{A},1}, \mathsf{view}_{\mathcal{A},2}, \widetilde{\mathsf{st}}, p) = (\mathsf{view}_{\mathcal{A},3}, \{\widetilde{w}_j\})$

   (e) If $\mathsf{Valid}(\mathsf{view}_{\mathcal{A},1}, \mathsf{view}_{\mathcal{A},2}, \mathsf{view}_{\mathcal{A},3}) = 0$ then set $i = i + 1$ and goto Item (b).

   (f) Otherwise, output $(\mathsf{ans}_i, \mathsf{view}_{\mathcal{A},1}, \mathsf{view}_{\mathcal{A},2}, \mathsf{view}_{\mathcal{A},3})$.

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

Claim 10. *Simulator $\mathcal{S}$ runs in expected polynomial time in λ.*

Proof. We analyze the runtime of each step of the simulation separately. Clearly, steps 1 and 2 takes only $\mathsf{poly}(\lambda)$ time. Denote by ϵ the probability with which the adversary $\mathcal{A}$ does not abort. Namely, ϵ is the probability that $\mathcal{S}^{\mathcal{A}}$ proceeds to Step 3, and the expected runtime of Step 3 (given that it is executed) is $\frac{\mathsf{poly}(\lambda)}{\epsilon}$. Thus, on average, this contributes another $\mathsf{poly}(\lambda)$ to the simulator's runtime.

We next argue that the expected runtime of Step 4 (given that it is executed) is $\mathsf{poly}(\lambda)/\epsilon$. Note that this step consists of running algorithm $\mathcal{E}_{\mathcal{A}}^1$ *once*, and then repeatedly running $\mathcal{E}_2^{\mathcal{A}}$ and $\mathcal{E}_3^{\mathcal{A}}$. Recall that the runtime of $\mathcal{E}_1^{\mathcal{A}}$ is $\frac{\mathsf{poly}(\lambda)}{p}$, whereas the runtime of $\mathcal{E}_2^{\mathcal{A}}$ and $\mathcal{E}_3^{\mathcal{A}}$ is $\mathsf{poly}(\lambda)$. Note that $\mathbb{E}[p] = \epsilon$. Thus, it suffices to prove that on expectation, the number of times that $\mathcal{E}_2^{\mathcal{A}}$ and $\mathcal{E}_3^{\mathcal{A}}$ are run (in Step 4) is $O(1/\epsilon)$. This follows from the fact that conditioned on $(\mathsf{view}_{\mathcal{A},1}, \mathsf{st})$ generated in Steps 1 and 2, the probability that $\mathcal{E}$ generates a valid transcript is $\Omega(\epsilon)$. This is the case since otherwise, it would contradict the zero-knowledge property of Definition 10.

We next analyze the expected runtime of Step 5 (given that it is executed). For each message $(\mathsf{view}_{\mathcal{A},2}, \widetilde{\mathsf{st}}_2)$ there is a different probability that the continuation will be valid. If for a given message $(\mathsf{view}_{\mathcal{A},2}, \widetilde{\mathsf{st}}_2)$ this probability is q, then the expected runtime of Step 5 is $\mathsf{poly}(\lambda)/q$. Since $\mathbb{E}[q] = \mathbb{E}[p] = \epsilon$, we conclude that the expected runtime of Step 5 is $\mathsf{poly}(\lambda)/\epsilon$, as desired.

The observation that $\mathbb{E}[q] = \epsilon$, immediately implies that Step 6 runs in expected $\frac{\lambda}{\epsilon}$ time. It thus remains to notice that Step 7 takes time $k \cdot \mathsf{poly}(\lambda)$, which together with the observation above, implies that the expected runtime is $\frac{\mathsf{poly}(\lambda)}{\epsilon}$, as desired.

We conclude that altogether, the expected runtime of Steps 3-7 is $\frac{\mathsf{poly}(\lambda)}{\epsilon}$. This, together with the fact that these steps are invoked with probability ϵ, implies that the overall expected runtime of $\mathcal{S}$ is $\epsilon \cdot \frac{\mathsf{poly}(\lambda)}{\epsilon} = \mathsf{poly}(\lambda)$, as desired.

□

We also note that by Definition 10, on completion of Step 4, the simulator correctly learns the adversary's randomness r. Definition 10 also guarantees that the adversary uses the same randomness when the simulator forces the output in Step 7, implying correctness of forced output.

Claim 11. *The ensembles* $\{\mathsf{REAL}_{\mathcal{A}}(1^\lambda, 1^\ell)\}_{\lambda\in\mathbb{N}}$ *and* $\{\mathsf{IDEAL}_{\mathcal{S}^{\mathcal{A}}}(1^\lambda, 1^\ell)\}_{\lambda\in\mathbb{N}}$ *are computationally indistinguishable.*

Proof. Suppose for contradiction that there exists a PPT distinguisher $\mathcal{D}$ and there exists a constant $c \in \mathbb{N}$ such that for infinitely many security parameters λ,

$$\left|\Pr[\mathcal{D}(\mathsf{REAL}_{\mathcal{A}}(1^\lambda, 1^\ell)) = 1] - \Pr[\mathcal{D}(\mathsf{IDEAL}_{\mathcal{S}^{\mathcal{A}}}(1^\lambda, 1^\ell)) = 1]\right| \geq \lambda^{-c}.$$

In what follows we get a contradiction, by considering a series of hybrids where the first hybrid Hyb_0 corresponds to the ideal world and the last hybrid Hyb_4 corresponds to the real world. We reach a contradiction by proving that

$$|\Pr[\mathcal{D}(\mathsf{Hyb}_4) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}_0) = 1]| < \lambda^{-c}. \tag{1}$$

- Hyb_0 **- Ideal World:** This hybrid is identical to the ideal world simulation $S^{\mathcal{A}}(1^\lambda, 1^\ell)$
- Hyb_1**:** This hybrid is similar to Hyb_0 except that in Step 4b, we choose $c_1 = \mathsf{Com}(0; s)$ (rather than choosing it with a random r_1), and yet Step 4e is executed w.r.t. a random $\mathsf{r}_1 \leftarrow \{0,1\}^\ell$.
- Hyb_2**:** This hybrid executes Hyb_1 until the end of Step 4, and outputs the view of $\mathcal{A}$ in this simulation.
- Hyb_3**:** This hybrid is similar to Hyb_2, except that rather than choosing p as in Step 3, it sets $p \triangleq \mathsf{pExtract}^{\mathcal{A}}_{2c}$, with respect to the distribution of instances (c, r), generated by sampling at random $r \leftarrow \{0,1\}^\ell$ and random $s \leftarrow \{0,1\}^\lambda$ and setting $c = \mathsf{Com}(r; s)$. If $p = 0$ then simply output $\perp$. Otherwise, execute Step 4 with this new value of p.
- Hyb_4 **- Real World:** In this hybrid, the simulator plays the role of the honest party (honestly) and emulates the malicious parties as instructed by $\mathcal{A}$. It outputs the view of $\mathcal{A}$.

We prove Equation 1, via the following sequence of claims.

Claim 12.

$$\mathsf{Hyb}_0 \approx \mathsf{Hyb}_1.$$

This claim follows immediately from the hiding property of the commitment scheme Com.

Claim 13.

$$\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_2.$$

This claim follows from the observation that Hyb_2 aborts in Step 7 only with negligible probability, and these hybrids are identical conditioned on the event that Hyb_2 does not abort in Step 7.

Claim 14. *For every PPT distinguisher* $\mathcal{D}$*,*

$$|\Pr[\mathcal{D}(\mathsf{Hyb}_2) = 1] - \Pr[\mathcal{D}(\mathsf{Hyb}_3) = 1]| \leq 2\lambda^{-2c} + \mathsf{negl}(\lambda).$$

This claim follows from the observation that if $\mathsf{pExtract}^{\mathcal{A}}_{2c} = 0$ then the probability that these hybrids abort immediately after Step 2 is at least $1 - 2\lambda^{-2c}$. Moreover, if $\mathsf{pExtract}^{\mathcal{A}}_{2c} > 0$, then by Definition 10, together with the fact that Step 3 (the validity approximation step) is a good approximation for the validity.

Claim 15.

$$\mathsf{Hyb}_3 \approx \mathsf{Hyb}_4.$$

This claim follows immediately from Definition 10. □

□

7 Four Round Malicious Secure MPC

7.1 Building Block: Extractable Commitments with Additional Properties

In this section, we describe a three round extractable commitment protocol $\mathsf{ECom} = (S, R)$. While several constructions of three round extractable commitment schemes are known in the literature (see, e.g., [PRS02, Ros04]), the commitment scheme we describe here achieves some stronger security properties that we describe below:

- *Bounded-Rewinding Security:* The commitment scheme satisfies a "bounded-rewinding security" property, which roughly means that the value committed by a sender in an execution of the commitment protocol remains hidden even if a malicious receiver can rewind the sender back to the start of the second round of the protocol an a priori bounded B number of times. In our application, we set $B = 4$; however, our construction also supports larger values of B.
- *Reusability:* The commitment scheme also satisfies a "reusability" property, which roughly means that the values committed by a sender in polynomially many executions of the commitment protocol remain hidden even if all of the executions share the same first two messages.

For technical reasons, we don't define or prove B-rewinding security property and reusability property for our extractable commitment protocol. Instead, this is done inline in the application - the four round MPC protocol in the next subsection.

Construction. Let Com denote a non-interactive perfectly binding commitment scheme based on injective one-way functions. Let N and B be positive integers such that $N - B - 1 \geq \frac{N}{2} + 1$. For $B = 4$, it suffices to set $N = 12$.

The three round extractable commitment protocol ECom is described in Figure 5.

Well-Formedness of ECom Transcripts. We now define a "well-formedness" property of an execution transcript of ECom. Roughly, we say that a transcript $(\mathsf{ECom}_1^{S\to R}, \mathsf{ECom}_2^{R\to S}, \mathsf{ECom}_3^{S\to R})$ is well-formed w.r.t. an input x and randomness r if:

- $N - 1$ out of the N tuples $\mathsf{ECom}_{3,\ell}^{S\to R} = (\alpha_\ell, \beta_\ell)$ (where $\ell \in [N]$) are "honestly" computed using randomness $r = \left(k, \{\mathsf{p}_i\}_{i=1}^N, \{r_i\}_{i=1}^N\right)$ in the sense that: each α_ℓ is a one-time pad of k w.r.t. the key $\mathsf{p}_\ell(0)$ where p_ℓ is a polynomial committed (using randomness r_ℓ) in the first round message $\mathsf{ECom}_1^{S\to R}$, and each β_ℓ is a correct evaluation of the polynomial p_ℓ over the "challenge" value z_ℓ contained in $\mathsf{ECom}_2^{R\to S}$.
- $\mathsf{ECom}_{3,N+2}^{S\to R}$ is a one-time pad of x w.r.t. the key $\mathsf{PRF}(k, \mathsf{ECom}_{3,N+1})$.

We now proceed to formally define the well-formedness property. For any set T, let $T[i]$ denote the i^{th} element of T.

Sender S has input x.

Commitment Phase:

1. **Round 1:**
   S does the following:
   - Pick N random degree B polynomials $\mathsf{p}_1, \ldots, \mathsf{p}_N$ over $\mathbb{Z}_q$, where q is a large prime.
   - Compute $\mathsf{ECom}_{1,\ell}^{S\to R} \leftarrow \mathsf{Com}(\mathsf{p}_\ell; r_\ell)$ using a random string r_ℓ, for every $\ell \in [N]$.
   - Send $\mathsf{ECom}_1^{S\to R} = (\mathsf{ECom}_{1,1}^{S\to R}, \ldots, \mathsf{ECom}_{1,N}^{S\to R})$ to R.
2. **Round 2:**
   R does the following:
   - Pick random values $\mathsf{z}_\ell \xleftarrow{\$} \mathbb{Z}_q$ for every $\ell \in [N]$.
   - Send $\mathsf{ECom}_2^{R\to S} = (\mathsf{z}_1, \ldots, \mathsf{z}_N)$ to S.
3. **Round 3:**
   S does the following:
   - Sample a PRF key k and a random string s.
   - Compute $\mathsf{ECom}_{3,\ell}^{S\to R} \leftarrow (k \oplus \mathsf{p}_\ell(0), \mathsf{p}_\ell(\mathsf{z}_\ell))$ for all $\ell \in [N]$.
   - Set $\mathsf{ECom}_{3,N+1}^{S\to R} = s$ and compute $\mathsf{ECom}_{3,N+2}^{S\to R} \leftarrow \mathsf{PRF}(k, s) \oplus x$.
   - Send $\mathsf{ECom}_3^{S\to R} = (\mathsf{ECom}_{3,1}^{S\to R}, \ldots, \mathsf{ECom}_{3,N+2}^{S\to R})$ to R.

Decommitment Phase:

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

   If any of the above verifications fail, R outputs $\perp$. Otherwise, R computes $x \leftarrow \mathsf{PRF}(k, \mathsf{ECom}_{3,N+1}) \oplus \mathsf{ECom}_{3,N+2}$.

Figure 5: Extractable Commitment Scheme ECom.

Definition 14 (Well-Formed Transcripts)**.** *An execution transcript* $(\mathsf{ECom}_1^{S\to R}, \mathsf{ECom}_2^{R\to S}, \mathsf{ECom}_3^{S\to R})$ *of* ECom *is said to be* well-formed *with respect to an input* x *and randomness* $r = \left(k, \{\mathsf{p}_i\}_{i=1}^N, \{r_i\}_{i=1}^N\right)$ *if there exists an index set* $\mathcal{I}$ *of size* $N-1$ *such that the following holds:*

- *For every* $j \in |\mathcal{I}|$, $\mathsf{ECom}_{1,\mathcal{I}[j]}^{S\to R} = \mathsf{Com}(\mathsf{p}_{\mathcal{I}[j]}; r_{\mathcal{I}[j]})$ *(AND)*
- *For every* $j \in |\mathcal{I}|$, $\mathsf{ECom}_{3,\mathcal{I}[j]}^{S\to R} = (k \oplus \mathsf{p}_\ell(0), \mathsf{p}_\ell(\mathsf{z}_{\mathcal{I}[j]}))$, *where* $\mathsf{ECom}_2^{R\to S} = (\mathsf{z}_1, \ldots, \mathsf{z}_N)$ *(AND)*
- $x = \mathsf{PRF}(k, \mathsf{ECom}_{3,N+1}) \oplus \mathsf{ECom}_{3,N+2}$.

We remark that the above well-formedness property is "weak" in the sense that we only require

$N-1$ out of the N tuples $\mathsf{ECom}_{3,\ell}^{S\to R} = (\alpha_\ell, \beta_\ell)$ to be honestly generated (instead of requiring that all N tuples are honestly generated). This relaxation is crucial to establishing the B-rewinding-security property for ECom. In fact, a slightly "trimmed" version of our construction where we do not use a PRF in the third round already suffices for achieving B-rewinding-security property. We use a PRF in the third round to achieve reusability property, in a similar manner to [JKKR17].

Extractor $\mathsf{Ext}_{\mathsf{ECom}}$. We describe a PPT extractor algorithm $\mathsf{Ext}_{\mathsf{ECom}}$ such that given a set of $(B+1)$ "well-formed" execution transcripts of ECom where each transcript consists the same first round sender message, the extractor successfully extracts the value committed in each transcript, except with negligible probability. We first define an "admissibility" property for any input to the extractor.

Definition 15 (Admissible Inputs)**.** *An input set* $(\mathsf{ECom}_1, \{\mathsf{ECom}_2^i, \mathsf{ECom}_3^i\}_{i=1}^{B+1})$ *is said to be* admissible *if for every* $i, j \in [B+1]$ *s.t.* $i \neq j$ *and every* $\ell \in [N]$*, we have that* $\mathsf{z}_\ell^i \neq \mathsf{z}_\ell^j$*, where* $\mathsf{ECom}_2^t = (\mathsf{z}_1^t, \ldots, \mathsf{z}_N^t)$.

The extractor algorithm $\mathsf{Ext}_{\mathsf{ECom}}$ is described in Figure 6.[16]

Input: An admissible set $(\mathsf{ECom}_1, \{\mathsf{ECom}_2^i, \mathsf{ECom}_3^i\}_{i=1}^{B+1})$ where $\forall i$, $(\mathsf{ECom}_1, \mathsf{ECom}_2^i, \mathsf{ECom}_3^i)$ is well-formed w.r.t. some value x_i.

1. For every $i \in [B]$, parse $\mathsf{ECom}_2^i = (\mathsf{z}_1^i, \ldots, \mathsf{z}_N^i)$ and $\mathsf{ECom}_3^i = (\mathsf{ECom}_{3,1}^i, \ldots, \mathsf{ECom}_{3,N+2}^i)$.
2. For each $\ell \in [N]$:
   - Parse $\mathsf{ECom}_{3,\ell}^i = (\alpha_\ell^i, \beta_\ell^i)$.
   - Using polynomial interpolation, compute a degree K polynomial p_ℓ over $\mathbb{Z}_q$ such that on point z_ℓ^i, $\mathsf{p}_\ell(\mathsf{z}_\ell^i) = \beta_\ell^i$.
   - Compute $k_\ell^i = (\alpha_\ell^i \oplus \mathsf{p}_\ell(0))$.
3. For every $i \in [B]$, let k^i be the value that equals a majority of the values in the set $\{k_1^i, \ldots, k_N^i\}$, and compute $x_i = \mathsf{PRF}(k^i, \mathsf{ECom}_{3,N+1}^i) \oplus \mathsf{ECom}_{3,N+2}^i$. If no such k^i value exists, set $x_i = \bot$.
4. Output $(x_1, \ldots, x_B)$.

Figure 6: Strategy of algorithm $\mathsf{Ext}_{\mathsf{ECom}}$.

We now analyze the extraction algorithm. Recall that for every $i \in [B+1]$, the transcript $(\mathsf{ECom}_1, \mathsf{ECom}_2^i, \mathsf{ECom}_3^i)$ is well-formed w.r.t. some value x_i. By the definition of well-formedness, we have that for every i, there exists at most one $j \in [N]$ such that $\mathsf{ECom}_{3,j}^i$ was not computed correctly and consistently with the other $\mathsf{ECom}_{3,j'}^i$. This means that overall, across all $i \in [B+1]$ execution transcripts, there exists at most $(B+1)$ values of $\mathsf{ECom}_{3,j}^i$ that were not computed correctly. This implies that for at least $(N-B-1)$ values of j, the values $\mathsf{ECom}_{3,j}^i$ were computed

[16] An admissible input set consisting of $(B+1)$ "well-formed" execution transcripts of ECom that share the same first round sender message can be obtained from a malicious sender via an expected PPT rewinding procedure. The expected PPT simulator in our application performs the necessary rewindings to obtain such transcripts and then feeds them to the extractor $\mathsf{Ext}_{\mathsf{ECom}}$.

correctly in all $B+1$ transcripts. This means that for every i, $(N-B-1)$ out of N values $\{k_1^i, \ldots, k_N^i\}$ computed by the extractor are the same. Then, since $N-B-1 \geq \frac{N}{2}+1$, we have that the extractor computes the correct values k^i and x_i for every $i \in [B]$.

7.2 The MPC Protocol

In this section, we describe our four round MPC protocol that achieves security against any malicious PPT adversary that corrupts a dishonest majority of parties. We obtain our result by "compiling" a three-round semi-malicious MPC protocol with some additional properties into a maliciously secure protocol. To formally state our result, we first recall the following result from [BHP17]:

Theorem 11. *[BHP17] Under the LWE assumption, there exists a 3-round protocol π^{SM} that has the following properties:*

1. *The first message of the protocol is public-coin (i.e., an honest party simply samples a random string and sends it as the first message).*
2. *The protocol π^{SM} is secure against semi-malicious adversaries, and the simulator of π^{SM} computes the first and second round messages exactly as the honest party algorithm with the input 0.*

We show how to generically transform any such protocol π^{SM} into a four round protocol π which is secure against malicious adversaries. Formally, we prove the following theorem.

Theorem 12. *Assuming:*

- *DDH or Quadratic Residuosity or N^{th} Residuosity assumptions, AND*
- *a three round MPC protocol π^{SM} for any functionality f that satisfies the properties of Theorem 11,*

the protocol π presented below is a four round MPC protocol for f.

Theorems 11 and 12 imply the following corollary.

Corollary 13. *Assuming:*

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

the protocol π presented below is a four round MPC protocol for any functionality f.

Below, we first list the ingredients used in our construction, followed by some relevant notation. We formally describe our MPC protocol in Section 7.3.

Ingredients. We list all the cryptographic ingredients that are used in our MPC protocol. As mentioned in the introduction, our construction uses a Promise ZK argument system in a non-black-box manner, and therefore we list all of its ingredients separately.

- NCom is a non-interactive commitment scheme based on injective one way functions.

- $\mathsf{WZK} = (\mathsf{WZK}_1, \mathsf{WZK}_2, \mathsf{WZK}_3, \mathsf{WZK}_4)$ is a three-message delayed-input distributional weak zero-knowledge argument system, where the algorithm WZK_4 is used to compute the decision of the verifier. Such a scheme was constructed by Jain et al.[JKKR17] based on DDH or Quadratic Residuosity or N^{th} Residuosity assumption.

- $\mathsf{ECom} = (\mathsf{ECom}_1, \mathsf{ECom}_2, \mathsf{ECom}_3, \mathsf{Ext}_{\mathsf{ECom}})$ is the three-message delayed-input extractable commitment scheme described in Section 7.1. We set the rewinding parameter B associated with ECom to be 4. Here $\mathsf{Ext}_{\mathsf{ECom}}$ is the extractor algorithm associated with ECom s.t. given an admissible input set consisting of 5 well-formed execution transcripts of the ECom protocol that share the same first round message, $\mathsf{Ext}_{\mathsf{ECom}}$ outputs all of the committed values with overwhelming probability.

- $\mathsf{TDGen} = (\mathsf{TDGen}_1, \mathsf{TDGen}_2, \mathsf{TDGen}_3, \mathsf{TDOut}, \mathsf{TDValid}, \mathsf{TDExt})$ is a three-message trapdoor generation protocol as defined in Section 4. The first three algorithms are used to generate the messages of the protocol while TDOut computes the receiver's output. Algorithm $\mathsf{TDValid}$ determines whether an input trapdoor value is valid w.r.t. the first message of a protocol transcript. Algorithm TDExt computes a valid trapdoor given three protocol transcripts that share the same first round message.

- $\mathsf{WI} = (\mathsf{WI}_1, \mathsf{WI}_2, \mathsf{WI}_3, \mathsf{WI}_4)$ is a three-message delayed-input witness-indistinguishable argument system, where WI_4 is used to compute the decision of the verifier. Such schemes are known from injective one-way functions [LS90].

- $\mathsf{RWI} = (\mathsf{RWI}_1, \mathsf{RWI}_2, \mathsf{RWI}_3, \mathsf{RWI}_4)$ is a three round delayed-input witness-indistinguishable argument with B-bounded rewinding security (for $B = 6$) as well as reusability security as defined in [JKKR17] (Definition 10).

  Such a protocol is described in Figure 6 in [JKKR17]; for our purposes, we will instantiate their protocol with the bounded rewinding secure WI from Appendix A. We note that the resulting protocol satisfies rewinding security in the setting where verifier challenges are non-adaptive, that is, the second message of the verifier for main thread of the WI is known before answering any rewinding queries. This rewinding security property suffices for our MPC construction.

- $\mathsf{NMCom} = (\mathsf{NMCom}_1, \mathsf{NMCom}_2, \mathsf{NMCom}_3)$ is the three-message special non-malleable commitment scheme of Goyal et al. [GPR16] satisfying Definition 4. Let $\mathsf{Ext}_{\mathsf{NMCom}}$ denote the PPT extractor associated with the 2-extraction property satisfied by NMCom.

- π^{SM} is the three-round semi-malicious MPC protocol from Theorem 11. 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 the completion of round i. Let $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ denote the straight line simulator for this protocol, where $\mathcal{S}_i$ is the simulator's algorithm to compute the i^{th} round messages.

Notation. Let $\mathsf{P}_1, \ldots, \mathsf{P}_n$ denote the n parties, and let λ denote the security parameter. We consider communication in the broadcast model where all of the protocol messages are sent over a broadcast channel. Below, we describe some additional notation:

- We augment our notation with $i \to j$ in the superscript to denote that a message is being sent by party P_i with party P_j as the intended recipient.

- The round number of any sub-protocol (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 ECom in our construction. We use a and b in the subscript to differentiate between the two.

NP Languages. In our construction, we use proofs for the following NP languages:

- Language L_1 is characterized by the following relation R_1:
  Statement : $\mathsf{st} = \mathsf{nc}$
  Witness : $\mathsf{w} = \mathsf{r}_{\mathsf{nc}}$
  $R_1(\mathsf{st}, \mathsf{w}) = 1$ if and only if $\mathsf{nc} = \mathsf{NCom}(1; \mathsf{r}_{\mathsf{nc}})$

  In our protocol, we use this language for delayed-input distributional WZK proofs. When restricting our attention to such proofs between parties P_i and P_j, where P_i is the prover and P_j is the verifier, we denote the corresponding language by $L_1^{i \to j}$.
- Language L_2 is characterized by the following relation R_2:
  Statement : $\mathsf{st} = \left(\{\mathsf{ecom}_{a,i}\}_{i=1}^3, \{\mathsf{ecom}_{b,i}\}_{i=1}^3, \mathsf{msg}_2, \mathsf{Trans}_1, \{\mathsf{nmcom}_i\}_{i=1}^3, \mathsf{td}_1, \mathsf{nc}\right)$
  Witness : $\mathsf{w} = (\mathsf{inp}, \mathsf{r}, \mathsf{r}_{a,\mathsf{ecom}}, \mathsf{r}_{b,\mathsf{ecom}}, \mathsf{t}, \mathsf{r}_{\mathsf{nmcom}}, \mathsf{r}_{\mathsf{nc}})$
  $R_2(\mathsf{st}, \mathsf{w}) = 1$ if and only if :

  1. Either $(\mathsf{ecom}_{a,1}, \mathsf{ecom}_{a,2}, \mathsf{ecom}_{a,3})$ or $(\mathsf{ecom}_{b,1}, \mathsf{ecom}_{b,2}, \mathsf{ecom}_{b,3})$ is a well-formed transcript of ECom w.r.t. input $(\mathsf{inp}, \mathsf{r})$ and randomness $\mathsf{r}_{a,\mathsf{ecom}}$ (see Definition 14), and msg_2 is an honestly computed second round message in protocol π^{SM} w.r.t. input inp and randomness r and round 1 protocol transcript Trans_1 (OR)
  2. $(\mathsf{nmcom}_1, \mathsf{nmcom}_2, \mathsf{nmcom}_3)$ is a transcript of a non-malleable commitment to a value t that is a valid trapdoor w.r.t. td_1. (OR)
  3. nc is a commitment to 0.

  Formally, $R_2(\mathsf{st}, \mathsf{w}) = 1$ if and only if :

  - $\mathsf{ecom}_{a,1} = \mathsf{ECom}_1(\mathsf{r}_{a,\mathsf{ecom}})$ AND
  - $\mathsf{ecom}_{a,3} = \mathsf{ECom}_3(\mathsf{inp}, \mathsf{r}, \mathsf{ecom}_{a,1}, \mathsf{ecom}_{a,2}; \mathsf{r}_{a,\mathsf{ecom}})$ AND
  - $\mathsf{msg}_2 = \pi_2^{\mathsf{SM}}(\mathsf{inp}, \mathsf{Trans}; \mathsf{r})$ AND
  - $(\mathsf{ecom}_{a,1}, \mathsf{ecom}_{a,2}, \mathsf{ecom}_{a,3})$ is well-formed w.r.t. input $(\mathsf{inp}, \mathsf{r})$ and randomness $\mathsf{r}_{a,\mathsf{ecom}}$

    (OR)

  - $\mathsf{ecom}_{b,1} = \mathsf{ECom}_1(\mathsf{r}_{b,\mathsf{ecom}})$ AND
  - $\mathsf{ecom}_{b,3} = \mathsf{ECom}_3(\mathsf{inp}, \mathsf{r}, \mathsf{ecom}_{b,1}, \mathsf{ecom}_{b,2}; \mathsf{r}_{b,\mathsf{ecom}})$ AND
  - $\mathsf{msg}_2 = \pi_2^{\mathsf{SM}}(\mathsf{inp}, \mathsf{Trans}_1; \mathsf{r})$ AND
  - $(\mathsf{ecom}_{b,1}, \mathsf{ecom}_{b,2}, \mathsf{ecom}_{b,3})$ is well-formed w.r.t. committed value $(\mathsf{inp}, \mathsf{r})$ and randomness $\mathsf{r}_{b,\mathsf{ecom}}$.

    (OR)

  - $\mathsf{TDValid}(\mathsf{td}_1, \mathsf{t}) = 1$ AND

– $\mathsf{nmcom}_1 = \mathsf{NMCom}_1(\mathsf{r}_{\mathsf{nmcom}})$ AND

– $\mathsf{nmcom}_3 = \mathsf{NMCom}_3(\mathsf{t}, \mathsf{nmcom}_1, \mathsf{nmcom}_2; \mathsf{r}_{\mathsf{nmcom}})$.

(OR)

– $\mathsf{nc} = \mathsf{NCom}(0; \mathsf{r}_{\mathsf{nc}})$.

In our protocol, we use language L_2 for bounded-rewinding secure delayed-input RWI proofs. When restricting our attention to such proofs between parties P_i and P_j, where P_i is the prover and P_j is the verifier, we denote the corresponding language by $L_2^{i\to j}$.

- Language L_3 is characterized by the following relation R_3:
  Statement : $\mathsf{st} = \left(\{\mathsf{ecom}_{a,i}\}_{i=1}^3, \{\mathsf{ecom}_{b,i}\}_{i=1}^3, \mathsf{msg}_3, \mathsf{Trans}_2, \{\mathsf{nmcom}_i\}_{i=1}^3, \mathsf{td}_1\right)$
  Witness : $\mathsf{w} = (\mathsf{inp}, \mathsf{r}, \mathsf{r}_{a,\mathsf{ecom}}, \mathsf{r}_{b,\mathsf{ecom}}, \mathsf{t}, \mathsf{r}_{\mathsf{nmcom}})$
  $R_3(\mathsf{st}, \mathsf{w}) = 1$ if and only if :

  1. Either $(\mathsf{ecom}_{a,1}, \mathsf{ecom}_{a,2}, \mathsf{ecom}_{a,3})$ or $(\mathsf{ecom}_{b,1}, \mathsf{ecom}_{b,2}, \mathsf{ecom}_{b,3})$ is a well-formed transcript of ECom w.r.t. input $(\mathsf{inp}, \mathsf{r})$ and randomness $\mathsf{r}_{a,\mathsf{ecom}}$ (see Definition 14), and msg_3 is an honestly computed third round message in protocol π^{SM} w.r.t. input inp and randomness r and round 2 protocol transcript Trans_2 (OR)
  2. $(\mathsf{nmcom}_1, \mathsf{nmcom}_2, \mathsf{nmcom}_3)$ is a transcript of a non-malleable commitment to a value t that is a valid trapdoor w.r.t. td_1.

  The formal description of R_3 is similar to R_2; we skip the details to avoid repetition.

  In our protocol, we use language L_3 for delayed-input WI proofs between parties. When restricting our attention to such proofs between parties P_i and P_j, where P_i is the prover and P_j is the verifier, we denote the corresponding language by $L_3^{i\to j}$.

7.3 The Protocol

We now proceed to formally describe our four round protocol π between n parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$. The input of party P_i is denoted as x_i.

Round 1: P_i does the following:

- Compute $\mathsf{msg}_{1,i} \leftarrow \pi_1^{\mathsf{SM}}(\mathsf{r}_i)$. Recall that π^{SM} is public-coin in the first round; thus, $\mathsf{msg}_{1,i}$ is a random string.
- Compute the first message of each of the three round delayed-input subprotocols. That is, for each $j \in [n]$ with $j \neq i$, P_i does the following:
  – Compute $\mathsf{ecom}_{a,1}^{i\to j} \leftarrow \mathsf{ECom}_1(\mathsf{r}_{a,\mathsf{ecom}}^{i\to j})$, $\mathsf{ecom}_{b,1}^{i\to j} \leftarrow \mathsf{ECom}_1(\mathsf{r}_{b,\mathsf{ecom}}^{i\to j})$, and $\mathsf{nmcom}_1^{i\to j} \leftarrow \mathsf{NMCom}_1(\mathsf{r}_{\mathsf{nmcom}}^{i\to j})$ using random strings $\mathsf{r}_{a,\mathsf{ecom}}^{i\to j}$, $\mathsf{r}_{b,\mathsf{ecom}}^{i\to j}$ and $\mathsf{r}_{\mathsf{nmcom}}^{i\to j}$, respectively.
  – Compute $\mathsf{td}_1^{i\to j} \leftarrow \mathsf{TDGen}_1(\mathsf{r}_{\mathsf{td}}^{i\to j})$ using a random string $\mathsf{r}_{\mathsf{td}}^{i\to j}$.
  – Compute $\mathsf{wi}_1^{i\to j} \leftarrow \mathsf{WI}_1(1^\lambda)$, $\mathsf{rwi}_1^{i\to j} \leftarrow \mathsf{RWI}_1(1^\lambda)$ and $\mathsf{wzk}_1^{i\to j} \leftarrow \mathsf{WZK}_1(1^\lambda)$.
- Broadcast $(\mathsf{msg}_{1,i}, \mathsf{ecom}_{a,1}^{i\to j}, \mathsf{ecom}_{b,1}^{i\to j}, \mathsf{nmcom}_1^{i\to j}, \mathsf{td}_1^{i\to j}, \mathsf{wi}_1^{i\to j}, \mathsf{rwi}_1^{i\to j}, \mathsf{wzk}_1^{i\to j})$.

Note that round 1 does not require the use of private input x_i.

Round 2: P_i now generates the second message of each of the three round delayed-input subprotocols. In more detail, for each $j \in [n]$ with $j \neq i$, P_i does the following:

- Compute $\mathsf{ecom}_{a,2}^{i\to j} \leftarrow \mathsf{ECom}_2(\mathsf{ecom}_{a,1}^{j\to i})$, $\mathsf{ecom}_{b,2}^{i\to j} \leftarrow \mathsf{ECom}_2(\mathsf{ecom}_{b,1}^{j\to i})$, and $\mathsf{nmcom}_2^{i\to j} \leftarrow \mathsf{NMCom}_2(\mathsf{nmcom}_1^{j\to i})$
- Compute $\mathsf{td}_2^{i\to j} \leftarrow \mathsf{TDGen}_2(\mathsf{td}_1^{j\to i})$.
- 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 $(\mathsf{ecom}_{a,2}^{i\to j}, \mathsf{ecom}_{b,2}^{i\to j}, \mathsf{nmcom}_2^{i\to j}, \mathsf{td}_2^{i\to j}, \mathsf{wi}_2^{i\to j}, \mathsf{rwi}_2^{i\to j}, \mathsf{wzk}_2^{i\to j})$.

Similar to round 1, round 2 also does not require the use of private input x_i.

Round 3: P_i does the following:

- Compute $\mathsf{msg}_{2,i} \leftarrow \pi_2^{\mathsf{SM}}(\mathsf{x}_i, \mathsf{Trans}_1; \mathsf{r}_i)$, where Trans_1 denotes the round 1 transcript of π^{SM}. (This is the first step where P_i uses its private input x_i.)
- Compute $\mathsf{nc}_i \leftarrow \mathsf{NCom}(1; \mathsf{r}_{\mathsf{nc},i})$.
- For each $j \in [n]$ with $j \neq i$, compute the following:
  - The third messages of the two extractable commitment schemes corresponding to inputs $(\mathsf{x}_i, \mathsf{r}_i)$ and $\perp$, respectively, and third message of the non-malleable commitment corresponding to input $\perp$. That is,

$$\mathsf{ecom}_{a,3}^{i\to j} \leftarrow \mathsf{ECom}_3((\mathsf{x}_i, \mathsf{r}_i), \mathsf{ecom}_{a,1}^{i\to j}, \mathsf{ecom}_{a,2}^{j\to i}; \mathsf{r}_{a,\mathsf{ecom}}^{i\to j}),$$

$$\mathsf{ecom}_{b,3}^{i\to j} \leftarrow \mathsf{ECom}_3(\perp, \mathsf{ecom}_{b,1}^{i\to j}, \mathsf{ecom}_{b,2}^{j\to i}; \mathsf{r}_{b,\mathsf{ecom}}^{i\to j}),$$

$$\mathsf{nmcom}_3^{i\to j} \leftarrow \mathsf{NMCom}_3(\perp, \mathsf{nmcom}_1^{i\to j}, \mathsf{nmcom}_2^{j\to i}; \mathsf{r}_{\mathsf{nmcom}}^{i\to j}),$$

    using random strings $\mathsf{r}_{a,\mathsf{ecom}}^{i\to j}$, $\mathsf{r}_{b,\mathsf{ecom}}^{i\to j}$, and $\mathsf{r}_{\mathsf{nmcom}}^{i\to j}$, respectively.
  - The third message of the trapdoor generation protocol, i.e., $\mathsf{td}_3^{i\to j} \leftarrow \mathsf{TDGen}_3(\mathsf{td}_1^{i\to j}, \mathsf{td}_2^{j\to i}; \mathsf{r}_{\mathsf{td}}^{i\to j})$ using randomness $\mathsf{r}_{\mathsf{td}}^{i\to j}$.
  - The third message of RWI, i.e., $\mathsf{rwi}_3^{i\to j} \leftarrow \mathsf{RWI}_3(\mathsf{rwi}_1^{i\to j}, \mathsf{rwi}_2^{j\to i}, \mathsf{st}_2^{i\to j}, \mathsf{w}_2^{i\to j})$ for the statement

$$\mathsf{st}_2^{i\to j} = \left(\{\mathsf{ecom}_{a,\ell}^{i\to j}\}_{\ell=1}^3, \{\mathsf{ecom}_{b,\ell}^{i\to j}\}_{\ell=1}^3, \mathsf{msg}_{2,i}, \mathsf{Trans}_1, \{\mathsf{nmcom}_\ell^{i\to j}\}_{\ell=1}^3, \mathsf{td}_1^{j\to i}, \mathsf{nc}_i\right) \in L_2^{i\to j}$$

    using witness $\mathsf{w}_2^{i\to j} = (\mathsf{x}_i, \mathsf{r}_i, \mathsf{r}_{a,\mathsf{ecom}}^{i\to j}, \perp, \perp, \perp, \perp)$.
  - The third message of the weak zero-knowledge protocol proving that nc_i is a commitment to 1. That is, $\mathsf{wzk}_3^{i\to j} \leftarrow \mathsf{WZK}_3(\mathsf{wzk}_1^{i\to j}, \mathsf{wzk}_2^{j\to i}, \mathsf{st}_1^{i\to j}, \mathsf{w}_1^{i\to j})$ for the statement $\mathsf{st}_1^{i\to j} = \mathsf{nc}_1^{i\to j} \in L_1^{i\to j}$ using witness $\mathsf{w}_1^{i\to j} = \mathsf{r}_{\mathsf{nc},i}$.
- Broadcast $(\mathsf{msg}_{2,i}, \mathsf{nc}_i, \mathsf{ecom}_{a,3}^{i\to j}, \mathsf{ecom}_{b,3}^{i\to j}, \mathsf{nmcom}_3^{i\to j}, \mathsf{td}_3^{i\to j}, \mathsf{rwi}_3^{i\to j}, \mathsf{wzk}_3^{i\to j})$.

Round 4: P_i does the following:

- Broadcast an abort signal to all the parties and abort if for any $j \in [n]$ s.t. $j \neq i$:
  - $\mathsf{TDOut}(\mathsf{td}_1^{j\to i}, \mathsf{td}_2^{i\to j}, \mathsf{td}_3^{j\to i}) \neq 1$ (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 $\mathsf{st}_1^{j\to i}$ is as defined above. (OR)
  - $\mathsf{RWI}_4(\mathsf{rwi}_1^{j\to i}, \mathsf{rwi}_2^{i\to j}, \mathsf{rwi}_3^{j\to i}, \mathsf{st}_2^{j\to i}) \neq 1$ where $\mathsf{st}_2^{j\to i}$ is as defined above.
- Compute $\mathsf{msg}_{3,i} \leftarrow \pi_3^{\mathsf{SM}}(\mathsf{x}_i, \mathsf{Trans}_2; \mathsf{r}_i)$, where Trans_2 denotes the round 2 transcript of π^{SM}.
- 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

$$\mathsf{st}_3^{i\to j} = (\{\mathsf{ecom}_{a,\ell}^{i\to j}\}_{\ell=1}^3, \{\mathsf{ecom}_{b,\ell}^{i\to j}\}_{\ell=1}^3, \mathsf{msg}_{3,i}, \mathsf{Trans}_2, \{\mathsf{nmcom}_\ell^{i\to j}\}_{\ell=1}^3, \mathsf{td}_1^{j\to i}) \in L_3^{i\to j}$$

  using witness $\mathsf{w}_3^{i\to j} = (\mathsf{x}_i, \mathsf{r}_i, \mathsf{r}_{a,\mathsf{ecom}}^{i\to j}, \bot, \bot, \bot)$.
- Broadcast $(\mathsf{msg}_{3,i}, \mathsf{wi}_3^{i\to j})$.

Output Computation: P_i does the following:

- Abort if for any $j \in [n]$ s.t. $j \neq i$:
  - $\mathsf{WI}_4(\mathsf{wi}_1^{j\to i}, \mathsf{wi}_2^{i\to j}, \mathsf{wi}_3^{j\to i}, \mathsf{st}_3^{j\to i}) \neq 1$ where $\mathsf{st}_3^{j\to i}$ is as defined above.
- Compute output $\mathsf{y}_i \leftarrow \mathsf{OUT}(\mathsf{x}_i, \mathsf{Trans}_3; \mathsf{r}_i)$, where Trans_3 denotes the round 3 transcript of π^{SM}.

This completes the description of protocol π.

7.4 Security Proof

In this section, we formally prove Theorem 12. Consider a malicious non-uniform PPT adversary $\mathcal{A}$ who corrupts $t < n$ parties. Let p be a polynomial such that $p(\lambda)$ denotes the total length of the input and randomness of each party P_i in protocol π^{SM}, i.e., $|(\mathsf{x}_i, \mathsf{r}_i)| = p(\lambda)$. We start by constructing an expected PPT black-box simulator Sim that simulates the view of $\mathcal{A}$.

7.4.1 Description of Simulator

Let us first recall the basic strategy followed by any black-box simulator who rewinds the adversary in order to simulate its view. Such a simulator creates a "main thread" of execution and a set of "look-ahead" threads. The main thread is the execution thread that is output at the end of the simulation, while the look-ahead threads facilitate the extraction of the adversary's inputs and some additional trapdoor information that is used to simulate the adversary's view on the main thread.

We construct an expected PPT black-box simulator Sim for protocol π. At a high-level, Sim works in the following two modes:

- **Case 1: Adversary does not abort in third round.** Suppose that the adversary does not abort in the third round of π. In this case, Sim rewinds the adversary in the second and third round of π to create look-ahead threads. Thus, 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 for each honest party. Sim uses 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 to simulate the main thread.

- **Case 2: Adversary aborts in third round.** In this case, Sim simply follows the honest party's strategy using input 0 on the main thread. It does not create any look-ahead threads.

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.

Simulator Sim. The simulator's description now follows:

Step 1 - Check Abort:

1. **Round 1:**
   For each honest party P_i, Sim follows the honest party algorithm in the first round. In more detail:
   - Compute $\mathsf{msg}_{1,i} \leftarrow \mathcal{S}_1(\mathsf{r}_i)$. Recall that similar to the honest party algorithm, this step simply consists of sampling a random string.
   - For each $j \in [n]$ with $j \neq i$, Sim does the following:
     - Compute $\mathsf{ecom}_{a,1}^{i\to j} \leftarrow \mathsf{ECom}_1(\mathsf{r}_{a,\mathsf{ecom}}^{i\to j})$, $\mathsf{ecom}_{b,1}^{i\to j} \leftarrow \mathsf{ECom}_1(\mathsf{r}_{b,\mathsf{ecom}}^{i\to j})$, and $\mathsf{nmcom}_1^{i\to j} \leftarrow \mathsf{NMCom}_1(\mathsf{r}_{\mathsf{nmcom}}^{i\to j})$ using random strings $\mathsf{r}_{a,\mathsf{ecom}}^{i\to j}$, $\mathsf{r}_{b,\mathsf{ecom}}^{i\to j}$ and $\mathsf{r}_{\mathsf{nmcom}}^{i\to j}$, 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 $(\mathsf{wzk}_1^{i\to j}) \leftarrow \mathsf{WZK}_1(1^\lambda)$.
   - Send $(\mathsf{msg}_{1,i}, \mathsf{ecom}_{a,1}^{i\to j}, \mathsf{ecom}_{b,1}^{i\to j}, \mathsf{nmcom}_1^{i\to j}, \mathsf{td}_1^{i\to j}, \mathsf{wi}_1^{i\to j}, \mathsf{rwi}_1^{i\to j}, \mathsf{wzk}_1^{i\to j})$ to $\mathcal{A}$.
2. **Round 2:**
   The simulator continues to follow the honest party algorithm in the second round. That is, for each honest party P_i and for each $j \in [n]$ with $j \neq i$, Sim does the following:
   - Compute $\mathsf{ecom}_{a,2}^{i\to j} \leftarrow \mathsf{ECom}_2(\mathsf{ecom}_{a,1}^{j\to i})$, $\mathsf{ecom}_{b,2}^{i\to j} \leftarrow \mathsf{ECom}_2(\mathsf{ecom}_{b,1}^{j\to i})$ and $\mathsf{nmcom}_2^{i\to j} \leftarrow \mathsf{NMCom}_2(\mathsf{nmcom}_1^{j\to i})$.
   - Compute $\mathsf{td}_2^{i\to j} \leftarrow \mathsf{TDGen}_2(\mathsf{td}_1^{j\to i})$.
   - 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 $\mathsf{wzk}_2^{i\to j} \leftarrow \mathsf{WZK}_2(\mathsf{wzk}_1^{j\to i})$.
   - Send $(\mathsf{ecom}_{a,2}^{i\to j}, \mathsf{ecom}_{b,2}^{i\to j}, \mathsf{nmcom}_2^{i\to j}, \mathsf{td}_2^{i\to j}, \mathsf{wi}_2^{i\to j}, \mathsf{rwi}_2^{i\to j}, \mathsf{wzk}_2^{i\to j})$ to $\mathcal{A}$.
3. **Round 3:**
   For each honest party P_i, Sim does the following:
   - Compute $\mathsf{msg}_{2,i} \leftarrow \mathcal{S}_2(\mathsf{Trans}_2; \mathsf{r}_i)$, where Trans_1 denotes the round 1 transcript of π^{SM}. Note that $\mathcal{S}_2$ simply executes the honest party algorithm with private input set to 0.
   - Compute $\mathsf{nc}_i \leftarrow \mathsf{Com}(1; \mathsf{r}_{\mathsf{nc},i})$.
   - For each $j \in [n]$ with $j \neq i$, compute:

– $\mathsf{ecom}_{a,3}^{i\to j} \leftarrow \mathsf{ECom}_3(0, \mathsf{r}_i, \mathsf{ecom}_{a,1}^{i\to j}, \mathsf{ecom}_{a,2}^{j\to i}; \mathsf{r}_{a,\mathsf{ecom}}^{i\to j})$, $\mathsf{ecom}_{b,3}^{i\to j} \leftarrow \mathsf{ECom}_3(\perp, \mathsf{ecom}_{b,1}^{i\to j}, \mathsf{ecom}_{b,2}^{j\to i}; \mathsf{r}_{b,\mathsf{ecom}}^{i\to j})$, and $\mathsf{nmcom}_3^{i\to j} \leftarrow \mathsf{NMCom}_3(\perp, \mathsf{nmcom}_1^{i\to j}, \mathsf{nmcom}_2^{j\to i}; \mathsf{r}_{\mathsf{nmcom}}^{i\to j})$ using random strings $\mathsf{r}_{a,\mathsf{ecom}}^{i\to j}$, $\mathsf{r}_{b,\mathsf{ecom}}^{i\to j}$ and $\mathsf{r}_{\mathsf{nmcom}}^{i\to j}$, respectively.

– $\mathsf{td}_3^{i\to j} \leftarrow \mathsf{TDGen}_3(\mathsf{td}_1^{i\to j}, \mathsf{td}_2^{j\to i}; \mathsf{r}_{\mathsf{td}}^{i\to j})$ using randomness $\mathsf{r}_{\mathsf{td}}^{i\to j}$.

– $\mathsf{rwi}_3^{i\to j} \leftarrow \mathsf{RWI}_3(\mathsf{rwi}_1^{i\to j}, \mathsf{rwi}_2^{j\to i}, \mathsf{st}_2^{i\to j}, \mathsf{w}_2^{i\to j})$ for the statement

$$\mathsf{st}_2^{i\to j} = \left(\{\mathsf{ecom}_{a,\ell}^{i\to j}\}_{\ell=1}^3, \{\mathsf{ecom}_{b,\ell}^{i\to j}\}_{\ell=1}^3, \mathsf{msg}_{2,i}, \mathsf{Trans}_1, \{\mathsf{nmcom}_\ell^{i\to j}\}_{\ell=1}^3, \mathsf{td}_1^{j\to i}, \mathsf{nc}_i\right) \in L_2^{i\to j}$$

using witness $\mathsf{w}_2^{i\to j} = (0, \mathsf{r}_i, \mathsf{r}_{a,\mathsf{ecom}}^{i\to j}, \perp, \perp, \perp, \perp)$.

– $\mathsf{wzk}_3^{i\to j} \leftarrow \mathsf{WZK}_3(\mathsf{wzk}_1^{i\to j}, \mathsf{wzk}_2^{j\to i}, \mathsf{st}_1^{i\to j}, \mathsf{w}_1^{i\to j})$ for the statement $\mathsf{st}_1^{i\to j} = \mathsf{nc}_1^{i\to j} \in L_1^{i\to j}$ using witness $\mathsf{w}_1^{i\to j} = (\mathsf{r}_{\mathsf{nc},i})$.

- Send $(\mathsf{msg}_{2,i}, \mathsf{nc}_i, \mathsf{ecom}_{a,3}^{i\to j}, \mathsf{ecom}_{b,3}^{i\to j}, \mathsf{nmcom}_3^{i\to j}, \mathsf{td}_3^{i\to j}, \mathsf{rwi}_3^{i\to j}, \mathsf{wzk}_3^{i\to j})$ to $\mathcal{A}$.

4. **Check Abort Condition:**
Simulator now checks whether $\mathcal{A}$ aborted in the third round. That is, for each honest party P_i, Sim does the following: for each $j \in [n]$ with $j \neq i$, set $\mathsf{flag} = \mathtt{abort}$ 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 $\mathsf{st}_2^{j\to i}$ is as defined above.

If $\mathsf{flag} = \mathtt{abort}$, then Sim outputs the partial view generated so far and stops. Otherwise, we will say that the "Check Abort" step succeeded. In this case, Sim proceeds to the next step.

Step 2 - Rewinding:

- Sim now rewinds $\mathcal{A}$ to the end of round 1. Then, Sim creates a set of T (defined later) look-ahead threads, where on each thread, only rounds 2 and 3 of the protocol are executed 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 executes the same strategy as in round 2 of step 1, using fresh randomness.
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 executes the same strategy as in round 3 of step 1, using fresh randomness.

- For each thread above, define it to be BAD if the "Check Abort" step fails (otherwise, we call it GOOD). That is, a thread is called 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 $\mathsf{st}_2^{j\to i}$ is as defined earlier.

- The number of threads T created is such that at least $(12 \cdot \lambda)$ GOOD threads exist. That is, Sim keeps running till it obtains $(12 \cdot \lambda)$ GOOD threads.

Step 3 - Input and Trapdoor Extraction:
Sim does the following:

- **Trapdoor Extraction:** For every pair $(\mathsf{P}_j, \mathsf{P}_i)$, where P_j is a corrupted party and P_i is an honest party, extract a trapdoor $\mathsf{t}^{j\to i}$ by running the trapdoor extractor TDExt on input the transcripts of the trapdoor generation protocol between P_j (playing the role of sender, a.k.a. trapdoor generator) and P_i (playing the role of receiver) from any 3 GOOD threads. That is, compute $\mathsf{t}^{j\to i} \leftarrow \mathsf{TDExt}\Big(\mathsf{td}_1^{j\to i}, \{\mathsf{td}_{2,\ell}^{i\to j}, \mathsf{td}_{3,\ell}^{j\to i}\}_{\ell=1}^{3}\Big)$, where $(\mathsf{td}_1^{j\to i}, \mathsf{td}_{2,\ell}^{i\to j}, \mathsf{td}_{3,\ell}^{j\to i})$ denotes the transcript of the trapdoor generation protocol between P_j and P_i on the ℓ^{th} GOOD thread.

- **Input Extraction:** Select any 5 GOOD threads, and any honest party P_i. For every corrupted party P_j, extract an input and randomness pair $(\mathsf{x}_{a,j}^{j\to i}, \mathsf{r}_{a,j}^{j\to i})$ by running the extractor $\mathsf{Ext}_{\mathsf{ECom}}$ on input the transcripts of the first extractable commitment protocol between P_j and P_i from the 5 GOOD threads. That is, compute:

$$(\mathsf{x}_{a,j}, \mathsf{r}_{a,j}) \leftarrow \mathsf{Ext}_{\mathsf{ECom}}\Big(\mathsf{ecom}_{a,1}^{j\to i}, \{\mathsf{ecom}_{a,2,\ell}^{i\to j}, \mathsf{ecom}_{a,3,\ell}^{j\to i}\}_{\ell=1}^{5}\Big),$$

  where $(\mathsf{ecom}_{a,1}^{j\to i}, \mathsf{ecom}_{a,2,\ell}^{i\to j}, \mathsf{ecom}_{a,3,\ell}^{j\to i})$ denotes the transcript of the first extractable commitment protocol between P_j and P_i on the ℓ^{th} GOOD thread. Similarly, compute $(\mathsf{x}_{b,j}^{j\to i}, \mathsf{r}_{b,j}^{j\to i})$ by running the extractor $\mathsf{Ext}_{\mathsf{ECom}}$ on input the transcripts of the second extractable commitment protocol between P_j and P_i from those 5 GOOD threads. Check which of $(\mathsf{x}_{a,j}, \mathsf{r}_{a,j})$ and $(\mathsf{x}_{b,j}, \mathsf{r}_{b,j})$ are consistent with the (partial) transcript of π^{SM} and set that pair to be $(\mathsf{x}_j, \mathsf{r}_j)$. Let R denote the set of all $\{\mathsf{x}_j, \mathsf{r}_j\}$ pairs extracted.

- 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 adversarial party P_j that was extracted in the previous step.

- Sim receives output y from the ideal functionality.

Step 5 - Abort Probability Estimation:
Set $\epsilon' = \frac{12\lambda}{T}$ as the probability with which the adversary doesn't abort (Recall that T is the number of threads created in Step 2).

Step 6 - Resampling the Main Thread:
First, Sim sets a counter value to 0. Sim now rewinds back to the end of round 1 and continues execution. (Note that this step also involves rewinding of its own 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 $\mathsf{t}^{j\to i}$ (extracted above). In more detail, for each honest P_i, Sim does the following:

   - Compute $\mathsf{msg}_{2,i} \leftarrow \mathcal{S}_2(\mathsf{Trans}_1; \mathsf{r}_i)$, where Trans_1 is the round 1 transcript of π^{SM}.

- Compute $\mathsf{nc}_i \leftarrow \mathsf{Com}(1; \mathsf{r}_{\mathsf{nc},i})$.
- For each $j \in [n]$ with $j \neq i$, compute:
  - Compute $\mathsf{ecom}_{a,3}^{i\to j} \leftarrow \mathsf{ECom}_3(0, \mathsf{r}_i, \mathsf{ecom}_{a,1}^{i\to j}, \mathsf{ecom}_{a,2}^{j\to i}; \mathsf{r}_{a,\mathsf{ecom}}^{i\to j})$, $\mathsf{ecom}_{b,3}^{i\to j} \leftarrow \mathsf{ECom}_3(\perp, \mathsf{ecom}_{b,1}^{i\to j}, \mathsf{ecom}_{b,2}^{j\to i}; \mathsf{r}_{b,\mathsf{ecom}}^{i\to j})$, and $\mathsf{nmcom}_3^{i\to j} \leftarrow \mathsf{NMCom}_3(\mathsf{t}^{j\to i}, \mathsf{nmcom}_1^{i\to j}, \mathsf{nmcom}_2^{j\to i}; \mathsf{r}_{\mathsf{nmcom}}^{i\to j})$ using random strings $\mathsf{r}_{a,\mathsf{ecom}}^{i\to j}$, $\mathsf{r}_{b,\mathsf{ecom}}^{i\to j}$ and $\mathsf{r}_{\mathsf{nmcom}}^{i\to j}$, respectively.
  - 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 $\mathsf{st}_2^{i\to j}$ as defined earlier, using witness $\mathsf{w}_2^{i\to j} = (0, \mathsf{r}_i, \mathsf{r}_{a,\mathsf{ecom}}^{i\to j}, \perp, \perp, \perp, \perp)$.
- Send $(\mathsf{msg}_{2,i}, \mathsf{nc}_i, \mathsf{ecom}_{a,3}^{i\to j}, \mathsf{ecom}_{b,3}^{i\to j}, \mathsf{nmcom}_3^{i\to j}, \mathsf{td}_3^{i\to j}, \mathsf{rwi}_3^{i\to j})$ to $\mathcal{A}$.

9. **Abort Condition:**
   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)
     - $\mathsf{RWI}_4(\mathsf{rwi}_1^{i\to j}, \mathsf{rwi}_2^{i\to j}, \mathsf{rwi}_3^{j\to i}, \mathsf{st}_2^{j\to i}) \neq 1$ where $\mathsf{st}_2^{j\to i}$ is as defined earlier.
   - If Sim's running time equals 2^λ, Abort.
   - If the counter value was not increased, continue to Step 7.
   - Else, if counter value is less than $\frac{\lambda^2}{\epsilon'}$, rewind back to the beginning of round 2 in Step 6. Abort otherwise.

Step 7 - Finishing the Main Thread:

10. **Round 4:**
    In the main thread, for each honest party P_i, Sim does the following:
    - Compute $\mathsf{msg}_{3,i} \leftarrow \mathcal{S}_3(\mathsf{y}, \mathsf{R}, \mathsf{Trans}_2, i)$, where Trans_2 denotes the round 2 transcript of protocol π^{SM}.
    - 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
      $$\mathsf{st}_3^{i\to j} = (\{\mathsf{ecom}_{a,\ell}^{i\to j}\}_{\ell=1}^3, \{\mathsf{ecom}_{b,\ell}^{i\to j}\}_{\ell=1}^3, \mathsf{msg}_{3,i}, \mathsf{Trans}_2, \{\mathsf{nmcom}_\ell^{i\to j}\}_{\ell=1}^3, \mathsf{td}_1^{j\to i}) \in L_3^{i\to j}$$
      using witness $\mathsf{w}_3^{i\to j} = (\perp, \perp, \perp, \perp, \mathsf{t}^{j\to i}, \mathsf{r}_{\mathsf{nmcom}}^{i\to j})$.
    - Send $(\mathsf{msg}_{3,i}, \mathsf{wi}_3^{i\to j})$ to $\mathcal{A}$.

11. **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 $\mathsf{st}_3^{j\to i}$ is as defined above. Also abort if any adversarial party aborted in the 4th round.

    In case of no abort, instruct the ideal functionality to deliver output to the honest parties.

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

Claim 16. *Simulator* Sim *runs in expected time that is polynomial in* λ.

Proof. We analyze the running time of each step performed by Sim:

- It is easy to see that Step 1 ("Check Abort") takes only $\mathsf{poly}(\lambda)$ time for some polynomial.
- Let ϵ be the probability with which "Check Abort" step succeeds. 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, Sim simply runs the extractors TDExt and $\mathsf{Ext}_{\mathsf{ECom}}$ that are both 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 :

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

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

This concludes the analysis. □

7.4.2 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 **- Determining Abort in 3rd Round:** 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, $\mathsf{Sim}_{\mathsf{Hyb}}$ executes the first 3 rounds of the protocol using the honest parties' strategy but using input 0. If the adversary does cause an abort, then $\mathsf{Sim}_{\mathsf{Hyb}}$ simply outputs the view of the adversary and stops. *In this case, the rest of the hybrids are skipped.*

  Otherwise, if the adversary doesn't cause an abort (i.e., 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, $\mathsf{Sim}_{\mathsf{Hyb}}$ does not use input 0 in the main thread that was done in the "Check Abort" step and instead continues using the honest parties' actual inputs. Further, if $\mathsf{Sim}_{\mathsf{Hyb}}$ receives an Abort at the end of round 3 in the main thread, it rewinds back to the end of round 1 and runs the main thread again. This process is repeated at most $\frac{1}{\mu}$ times. Observe that $\mathsf{Sim}_{\mathsf{Hyb}}$ runs in polynomial time because, by assumption, μ was non-negligible. (The same argument will hold in the subsequent hybrids as well.)

  The remaining hybrids are only considered if the "Check Abort" step succeeded.

- Hyb_2 **- Input and Trapdoor Extraction:** $\mathsf{Sim}_{\mathsf{Hyb}}$ creates a set of $\frac{5\cdot\lambda}{\mu}$ look-ahead threads in a similar manner as described in Step 2 of Sim. 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 and Trapdoor 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 function 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 on main thread:** 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 $\mathsf{nmcom}_3^{i\to j} \leftarrow \mathsf{NMCom}_3(\mathsf{t}^{j\to i}, \mathsf{nmcom}_1^{i\to j}, \mathsf{nmcom}_2^{j\to i}; \mathsf{r}_{\mathsf{nmcom}}^{i\to j})$ where $\mathsf{t}^{j\to i}$ is a valid trapdoor extracted as in the previous hybrid. That is, the simulator now commits to a trapdoor $\mathsf{t}^{j\to i}$ in the non-malleable commitment between P_i (committer) and P_j (receiver).

  Note: The above change 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 input and trapdoor extraction. The same argument applies to each subsequent hybrid as well when we make changes on the main thread.

- Hyb_4 **- Switching RWI proofs on main thread:** 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

$$\mathsf{st}_2^{i\to j} = \left(\{\mathsf{ecom}_{a,\ell}^{i\to j}\}_{\ell=1}^3, \{\mathsf{ecom}_{b,\ell}^{i\to j}\}_{\ell=1}^3, \mathsf{msg}_{2,i}, \mathsf{Trans}_1, \{\mathsf{nmcom}_\ell^{i\to j}\}_{\ell=1}^3, \mathsf{td}_1^{j\to i}, \mathsf{nc}_i\right) \in L_2^{i\to j}$$

  using witness $\mathsf{w}_2^{i\to j} = (\bot, \bot, \bot, \bot, \mathsf{t}^{j\to i}, \mathsf{r}_{\mathsf{nmcom}}^{i\to j}, \bot)$.
  That is, in the main thread, the simulator now uses the "trapdoor" witness (which establishes that it committed to a valid trapdoor in the non-malleable commitment) to compute the last message of each RWI proof where an honest party plays the role of the prover.

- Hyb_5 **- Switching WI proofs on main thread:** 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

$$\mathsf{st}_3^{i\to j} = (\{\mathsf{ecom}_{a,\ell}^{i\to j}\}_{\ell=1}^3, \{\mathsf{ecom}_{b,\ell}^{i\to j}\}_{\ell=1}^3, \mathsf{msg}_{3,i}, \mathsf{Trans}_2, \{\mathsf{nmcom}_\ell^{i\to j}\}_{\ell=1}^3, \mathsf{td}_1^{j\to i}) \in L_3^{i\to j}$$

using witness $\mathsf{w}_3^{i\to j} = (\bot, \bot, \bot, \bot, \mathsf{t}^{j\to i}, \mathsf{r}_{\mathsf{nmcom}}^{i\to j})$.
That is, in the main thread, the simulator now uses the "trapdoor" witness (which establishes that it committed to a valid trapdoor in the non-malleable commitment) to compute the last message of each WI proof where an honest party plays the role of the prover.

- Hyb_6 **- Changing EComs on main thread:** 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 $\mathsf{ecom}_{a,3}^{i\to j} = \mathsf{ECom}_3(0, \mathsf{r}_i, \mathsf{ecom}_{a,1}^{i\to j}, \mathsf{ecom}_{a,2}^{j\to i}; \mathsf{r}_{a,\mathsf{ecom}}^{i\to j})$.
That is, in the main thread, the simulator now commits to input 0 in the extractable commitment scheme.

- Hyb_7 **- Simulate π^{SM} on main thread:** 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, and Trans_1 denotes the transcript of protocol π^{SM} after round 1.
  - 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 on main thread:** 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 $\mathsf{st}_2^{i\to j}$ as defined earlier, using witness $\mathsf{w}_2^{i\to j} = (0, \mathsf{r}_i, \mathsf{r}_{a,\mathsf{ecom}}^{i\to j}, \bot, \bot, \bot, \bot)$.
That is, in the main thread, the simulator now uses the "honest" witness to compute the last message of each RWI proof where an honest party plays the role of the prover.

- Hyb_9 **- Extra look-ahead threads:** In this hybrid, $\mathsf{Sim}_{\mathsf{Hyb}}$ creates another set of $\frac{5\cdot\lambda}{\mu}$ look-ahead threads that only consist of rounds 2 and 3 (as in the first set of look-ahead threads). 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 $\{\mathsf{t}^{j\to i}, \mathsf{x}_j, \mathsf{r}_j\}$ using the first set of threads T_1.

  $\mathsf{Sim}_{\mathsf{Hyb}}$ also runs the "input and trapdoor extraction phase" using this second set of look-ahead threads. That is, $\mathsf{Sim}_{\mathsf{Hyb}}$ also extracts values $\{\tilde{\mathsf{t}}^{j\to i}, \tilde{\mathsf{x}}_j, \tilde{\mathsf{r}}_j\}$ 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 $(\mathsf{x}_j, \mathsf{r}_j) \neq (\tilde{\mathsf{x}}_j, \tilde{\mathsf{r}}_j)$ in any of the second set of look-ahead threads. $\mathsf{Sim}_{\mathsf{Hyb}}$ continues to use the first set of extracted values in simulating the main thread.

- Hyb_{10} **- Changes to extra set of look-ahead threads:** 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. (Note that this hybrid can be expanded into 6 hybrids for each thread in T_2.)

- Hyb_{11} **- Stop look-aheads in** T_1**:** $\mathsf{Sim}_{\mathsf{Hyb}}$ now stops executing the first set of look-ahead threads T_1 and uses the extracted values $\{\tilde{\mathsf{t}}_j, \tilde{\mathsf{x}}_j, \tilde{\mathsf{r}}_j\}$ from the second set of look-ahead threads for simulating the main thread.

  Further, the number of look-ahead threads in T_2 is increased from $\frac{5\cdot\lambda}{\mu}$ to as many as needed to estimate the probability of the adversary not aborting - ϵ'. That is, $\mathsf{Sim}_{\mathsf{Hyb}}$ performs steps 2-5 exactly as done in the ideal world.

  Additionally, at this point, $\mathsf{Sim}_{\mathsf{Hyb}}$ doesn't rewind the main thread $\frac{1}{\mu}$ times. Instead, $\mathsf{Sim}_{\mathsf{Hyb}}$ runs the rewound main thread for $\mathsf{min}(2^\lambda, \frac{\lambda^2}{\epsilon'})$ time as in the ideal world. This hybrid exactly corresponds to the ideal world.

7.4.3 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 16 (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 tuple* $(\mathsf{nmcom}_1^{j\to i}, \mathsf{nmcom}_2^{i\to j}, \mathsf{nmcom}_3^{j\to i})$ *denote the messages of the non-malleable commitment with* P_j *as the committer.*
Consider the following event E *which occurs if* $\exists(j, i, \mathsf{t}^{i\to j}, \mathsf{r}_{\mathsf{nmcom}}^{j\to i})$ *such that:*

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

That is, the event E *occurs if any corrupted party* P_j *commits to a valid trapdoor* $\mathsf{t}^{i\to j}$ *(corresponding to the trapdoor generation protocol between* P_i *and* P_j *where* P_i *was the trapdoor generator) in the non-malleable commitment protocol with* P_i*.*

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

Claim 17. *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 18. *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 19. *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 B 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 20. *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 21. *Assuming soundness of the argument systems* $\mathsf{WI}, \mathsf{RWI}$ *and* WZK*, the existence of an extractor* Ext *for the extractable commitment scheme* 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 $\mathsf{Ext}_{\mathsf{ECom}}$ 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. Then, by Markov's inequality, if we run $\frac{5 \cdot \lambda}{\mu}$ threads, except with negligible probability, at least 5 look-ahead threads will be "Good" threads.

Now, let's show that the algorithm $\mathsf{Ext}_{\mathsf{ECom}}$ successfully extracts except with negligible 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 $(\mathsf{ecom}_{a,1}^{j \to i}, \mathsf{ecom}_{a,2}^{i \to j}, \mathsf{ecom}_{a,3}^{j \to i})$ or the values $(\mathsf{ecom}_{b,1}^{j \to i}, \mathsf{ecom}_{b,2}^{i \to j}, \mathsf{ecom}_{b,3}^{j \to i})$ form a "well-formed" tuple of the scheme ECom as defined in Section 7.1.

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

Since both TDExt and Ext are successful except with negligible 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 a similar argument in every subsequent hybrid when we argue that if we create just 5 look-ahead threads, they are all "Good" with noticeable probability and hence we can extract the adversary's input and trapdoor with noticeable probability. □

Claim 22. *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 nmcom_1^L on the left side which is set as $\mathsf{nmcom}_1^{i\to j}$ in its interaction with $\mathcal{A}$ in round 1 of protocol π. On receiving $\mathsf{nmcom}_1^{j\to i}$, $\mathcal{A}_{\mathsf{NMCom}}$ 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 $\mathsf{nmcom}_3^{i\to j}$ as a commitment to $\perp$. Recall from the definition of the scheme NMCom that $\mathcal{A}_{\mathsf{NMCom}}$ can do this even without knowing the randomness used to generate $\mathsf{nmcom}_1^{i\to j}$. As in the proof of Claim 21, 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 nmcom_2^R from $\mathcal{C}_{\mathsf{NMCom}}$ as the second round message on the right side which it sets as the value $\mathsf{nmcom}_2^{i\to j}$ in round 2 of protocol π on the main thread. Then, on receiving $\mathsf{nmcom}_2^{j\to i}$ 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 $(\perp, \mathsf{t}^{j\to i})$ where $\mathsf{t}^{j\to i}$ is generated in the input extraction phase. Recall that NMCom is a delayed-input scheme.

$\mathcal{A}_{\mathsf{NMCom}}$ receives a third round message nmcom_3^L which is either a commitment to $\perp$ or $\mathsf{t}^{j\to i}$. This is sent to $\mathcal{A}$ as the value $\mathsf{nmcom}_3^{i\to j}$ in the main thread and the rest of protocol π is performed exactly as in Hyb_1. On receiving the value $\mathsf{nmcom}_3^{i\to j}$ 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 $\perp$ 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 $\mathsf{nmcom}_1^{i\to j}$.

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 23. *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 22 above.

□

Claim 24. *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

$$\mathsf{st}_2^{i\to j} = \left(\{\mathsf{ecom}_{a,\ell}^{i\to j}\}_{\ell=1}^3, \{\mathsf{ecom}_{b,\ell}^{i\to j}\}_{\ell=1}^3, \mathsf{msg}_{2,i}, \mathsf{Trans}_1, \{\mathsf{nmcom}_\ell^{i\to j}\}_{\ell=1}^3, \mathsf{td}_1^{j\to i}, \mathsf{nc}_i\right) \in L_2^{i\to j}$$

where the other values are generated as in Hyb_3.

In the main thread, $\mathcal{A}_{\mathsf{RWI}}$ also sends the pair of witnesses $(\mathsf{x}_i, \mathsf{r}_i, \mathsf{r}_{\mathsf{ecom}}^{i\to j}, \bot, \bot, \bot, \bot)$ and $(\bot, \bot, \bot, \bot, \mathsf{t}^{j\to i}, \mathsf{r}_{\mathsf{nmcom}}^{i\to j}, \bot)$ where $\mathsf{t}^{j\to i}$ is generated in the input extraction phase. For each look-ahead thread, $\mathcal{A}_{\mathsf{RWI}}$ sends the witness $(\mathsf{x}_i, \mathsf{r}_i, \mathsf{r}_{\mathsf{ecom}}^{i\to j}, \bot, \bot, \bot, \bot)$. 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 21 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 B 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 25. *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 24 above.

□

Claim 26. *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 27. *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 25 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 28. *If the "Check Abort" step succeeds, assuming the bounded rewinding security of the* RWI *, the hiding property of the commitment scheme* Com *and the pseudorandom function* PRF *used in the construction of* 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 $(\mathsf{x}_i, \mathsf{r}_i)$. We drop the subscript "a" throughout the description of this proof.

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 ECom from Section 7.1 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 $\mathsf{ecom}_1^{i\to j} = \{\mathsf{ecom}_{1,1}^{i\to j}, \ldots, \mathsf{ecom}_{1,N}^{i\to j}\}$ where $\mathsf{ecom}_{1,\ell}^{i\to j} = \mathsf{Com}(\mathsf{p}_\ell^{i\to j})$ where each $\mathsf{p}_\ell^{i\to j}$ is a random polynomial of degree 4 (defined in Section 7.1). In round 2, P_j generates $\mathsf{ecom}_2^{j\to i} = (\mathsf{z}_1^{j\to i}, \ldots, \mathsf{z}_N^{j\to i})$ where each $\mathsf{z}_\ell^{j\to i}$ is a random value. Then, in round 3, P_i outputs $\mathsf{ecom}_3^{i\to j} = \{\mathsf{ecom}_{3,1}^{i\to j}, \ldots, \mathsf{ecom}_{3,N+2}^{i\to j}\}$ where $\mathsf{ecom}_{3,\ell}^{i\to j} = (k^{i\to j} \oplus \mathsf{p}_\ell^{i\to j}(0), \mathsf{p}_\ell^{i\to j}(\mathsf{z}_\ell^{j\to i}))$ for each $\ell \in [N]$ and $\mathsf{ecom}_{3,N+1} = s^{i\to j}$ and $\mathsf{ecom}_{3,N+2} = \mathsf{PRF}(k^{i\to j}, s^{i\to j}) \oplus m$.

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 $\mathsf{ecom}_{3,\ell}^{i\to j} = (0 \oplus \mathsf{p}_\ell^{i\to j}(0), \mathsf{p}_\ell^{i\to j}(\mathsf{z}_\ell^{j\to i}))$ for all $\ell \in [N]$. 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 $\mathsf{ecom}_{3,N+2}^{i\to j}$ 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 $\mathsf{ecom}_{3,N+2}^{i\to j} = \mathsf{PRF}(k^{i\to j}, \mathsf{ecom}_{3,N+1}^{i\to j}) \oplus 0$.
- $\mathsf{Sub.Hyb}_5$**:** In this hybrid, for every honest party P_i and malicious party P_j, in the main thread, compute $\mathsf{ecom}_{3,\ell}^{i\to j} = (k^{i\to j} \oplus \mathsf{p}_\ell^{i\to j}(0), \mathsf{p}_\ell^{i\to j}(\mathsf{z}_\ell^{j\to i}))$ for all $\ell \in [N]$. 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* 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 $\mathsf{Sub.Hyb}_1$, for all $\ell \in [N]$, in $\mathsf{ecom}_{3,\ell}^{i\to j}$ we use $k^{i\to j}$ while in $\mathsf{Sub.Hyb}_2$, we use 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 $\mathsf{Sub.Hyb}_{1,N}$ where $\mathsf{Sub.Hyb}_{1,0}$ denotes $\mathsf{Sub.Hyb}_1$ and $\mathsf{Sub.Hyb}_{1,N}$ denotes $\mathsf{Sub.Hyb}_2$. We will then show that the invariant holds in every intermediate hybrid to complete the proof. **For** $\ell = 1 \ldots N$**,** $\mathsf{Sub.Hyb}_{1,\ell}$**:** 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 $(k \oplus \mathsf{p}_\ell(0)) = (0 \oplus \mathsf{q}_\ell(0))$.
- In round 1, compute $\mathsf{ecom}_{1,\ell} = \mathsf{Com}(\mathsf{q}_\ell)$.
- in the main thread, compute $\mathsf{ecom}_{3,\ell}$ as $(0 \oplus \mathsf{q}_\ell(0), \mathsf{q}_\ell(\mathsf{z}_\ell))$.
- In every look-ahead thread, compute $\mathsf{ecom}_{3,\ell}$ as before, i.e using input k but using polynomial q_ℓ.

That is, in $\mathsf{Sub.Hyb}_{1,\ell}$, we switch the ℓ^{th} index of the extractable commitment scheme from using input (k) 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 $\mathsf{Sub.Hyb}_{1,\ell}$ but holds in $\mathsf{Sub.Hyb}_{1,0}, \ldots, \mathsf{Sub.Hyb}_{1,\ell-1}$. 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 $\mathsf{Sub.Hyb}_{1,\ell}$. 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 $\mathsf{Sub.Hyb}_{1,\ell-1}$. Consider a set of hybrids $\mathsf{H}_1, \ldots, \mathsf{H}_5$ as follows where H_1 corresponds to $\mathsf{Sub.Hyb}_{1,\ell-1}$ and H_5 corresponds to $\mathsf{Sub.Hyb}_{1,\ell}$.

- 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 $(\mathsf{ecom}_1, \mathsf{ecom}_2, \mathsf{ecom}_3)$ is "well-formed" using all indices from $1, \ldots, \lambda$ except index ℓ.
- H_3**:** In round 1, for every honest party P_i and malicious party P_j, compute $\mathsf{ecom}_{1,\ell} = \mathsf{Com}(0)$.
- H_4**:** For every honest party P_i and malicious party P_j:
  - In the main thread, compute $\mathsf{ecom}_{3,\ell}$ as $(0 \oplus \mathsf{q}_\ell(0), \mathsf{q}_\ell(\mathsf{z}_\ell))$.
  - In every look-ahead thread, compute $\mathsf{ecom}_{3,\ell}$ as before but using polynomial q_ℓ.
- H_5**:** In round 1, for every honest party P_i and malicious party P_j, compute $\mathsf{ecom}_{1,\ell} = \mathsf{Com}(\mathsf{q}_\ell)$.

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

$$\mathsf{st}_2^{i\to j} = \left(\{\mathsf{ecom}_{a,\ell}^{i\to j}\}_{\ell=1}^3, \{\mathsf{ecom}_{b,\ell}^{i\to j}\}_{\ell=1}^3, \mathsf{msg}_{2,i}, \mathsf{Trans}_1, \{\mathsf{nmcom}_\ell^{i\to j}\}_{\ell=1}^3, \mathsf{td}_1^{j\to i}, \mathsf{nc}_i\right) \in L_2^{i\to j}$$

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 21 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 $\mathsf{ecom}_{1,\ell}$ in round 1 as $\mathsf{ecom}_{1,\ell} = \mathsf{Com}(0)$ 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 nmcom which is sent to $\mathcal{A}$ as the value $\mathsf{ecom}_{1,\ell}$ round 1 and the rest of protocol π is performed exactly as in H_2. In particular, recall from the proof of Claim 21 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 $\mathsf{ecom}^{i\to j}_{3,N+2}$ is computed uniformly at random while in $\mathsf{Sub.Hyb}_2$, it was computed as $\mathsf{PRF}(k^{i\to j}, \mathsf{ecom}^{i\to j}_{3,N+1}) \oplus \mathsf{x}_i$. 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* 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 28. □

Claim 29. *Assuming the bounded rewinding security of the scheme* RWI *and the hiding property of the underlying commitment scheme* Com *used in the construction of* ECom*,* Hyb_5 *is indistinguishable from* Hyb_6.

Proof. This is similar to the proof of Claim 28.

□

Claim 30. *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 21, 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 31. *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 30 above.

□

Claim 32. *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 24. □

Claim 33. *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 25. □

Claim 34. *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 35. *Assuming soundness of the argument systems* $\mathsf{WI}, \mathsf{RWI}$ *and* WZK*, the existence of an extractor* Ext *for the extractable commitment scheme* 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 19 : $\mathsf{Pr}[$ $\mathsf{Sim}_{\mathsf{Hyb}}$ 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 $\{\mathsf{x}_j, \mathsf{r}_j\} \neq \{\tilde{\mathsf{x}}_j, \tilde{\mathsf{r}}_j\}$. 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, $\{\mathsf{x}_j, \mathsf{r}_j\} = \{\tilde{\mathsf{x}}_j, \tilde{\mathsf{r}}_j\}$ and this completes the proof. □

Claim 36. *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. The difference between the two hybrids is that, in each of the second set of the look-ahead threads, one by one, $\mathsf{Sim}_{\mathsf{Hyb}}$ makes the same set of changes as done on the main thread from Hyb_3 to Hyb_8. Thus, this proof follows by a repeated application of the proofs of Claim 22, Claim 23, Claim 24, Claim 25, Claim 26, Claim 27, Claim 28, Claim 29, Claim 30, Claim 31, Claim 32, Claim 33. □

Claim 37. 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. Also, in both hybrids, if the adversary's abort probability is non-negligible, $\mathsf{Sim}_{\mathsf{Hyb}}$ correctly extracts the input with overwhelming probability (that is, $\mathsf{Sim}_{\mathsf{Hyb}}$ gets atleast 5 "Good" look-ahead threads in both hybrids).

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. □

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.

[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.

[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.

[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.

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

[BHP17] Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. In *TCC*, 2017.

[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.

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

[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.

[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.

[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.

[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.

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

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

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

[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.

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

[GGJS12] Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Concurrently secure computation in constant rounds. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 99–116, 2012.

[GJO10] Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Password-authenticated session-key generation on the internet in the plain model. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 277–294, 2010.

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

[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.

[GMPP16] Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. In *EUROCRYPT*, pages 448–476, 2016.

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

[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.

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

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

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

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

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

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

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

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

[JKKR17] Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Ron Rothblum. Distinguisher-dependent simulation in two rounds and its applications. In *CRYPTO*, 2017.

[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.

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

[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.

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

[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.

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

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

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

[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.

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

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

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

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

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

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

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

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

A WI with Bounded Rewinding Security

In this section, we sketch the proof of Theorem 6.

More generally, for any constant $B > 0$, we construct a 3 round delayed-input witness indistinguishable argument with B-rewinding security according to Definition 6 assuming the existence of injective one way functions.

Overview of the Protocol and Proof. 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 protocol of [IKOS07]. The 3-round delayed-input WI protocol in [LS90] for proving that a Graph G contains Hamiltonian cycle H is as follows. The prover first commits to a random hamiltonian matrix H' in the first round. Upon receiving verifier challenge, if this challenge is 0, the prover decommits the entire matrix H' (containing 0's representing non-edges and 1's representing edges), and the verifier checks if it is Hamiltonian. If this challenge is 1, the prover uses input G, H to compute a random permutation π that maps H to H'. It outputs π, and additionally decommits all values in H' that correspond to non-edges in $\pi(G)$. The verifier checks that all opened values in H' are 0.

The main idea to obtain bounded rewinding security, is to offload the process of verifying the commitment to the Hamiltonian H' (corresponding to $b = 0$), to an "MPC-in-the-Head" protocol [IKOS07] with bounded rewinding security. Opening this up, the commitment $c = \mathsf{Com}(H')$ will be accompanied by commitments to views of $n = (6B + 1)$ virtual players in a *robust* MPC protocol, with privacy against any subset of $2B$ collusions. The MPC protocol will output 1 to all virtual players iff H' is indeed a commitment to a hamiltonian. The verifier challenge corresponding to $b = 0$, will be replaced by $\binom{n}{2}$ possible challenges revealing the views of different subsets of 2 players, and the verifier will check these views for correctness and consistency. By $2B$-privacy of MPC, even given the views of $2B$ players, a malicious verifier will learn no information about H', giving us the desired bounded-rewinding security. Soundness will follow by [LS90] because the soundness of MPC-in-the-head will guarantee that H' is Hamiltonian.

To prove B-rewinding security, we note that each query of the malicious verifier V^* can be interpreted as consisting of two parts - the first is a bit b that is either 0 or 1 and if $b = 0$, the second part consists of a random subset of 2 parties whose views must be opened. The simulator runs by guessing the set of queries made across all the B-rewinds in advance, and appropriately running the [LS90] and [IKOS07] simulators based on this guess. We now describe the construction for completeness.

MPC-in-the-Head. We make black-box use of a 3-round zero knowledge protocol (non delayed-input) with constant soundness error and bounded rewinding security.

Definition 17 (3-Round Constant Soundness ZK with Bounded Rewinding Security)**.** *Fix a positive integer B. A delayed-input 3-round interactive argument (as defined in Definition 1) for an NP language L, with an NP relation R_L is said to have B-*Rewinding Security *if there exists a simulator* Sim *such that for every non-uniform PPT interactive Turing Machine V^*, it holds that $\{\mathsf{REAL}^{V^*}(1^\lambda)\}_\lambda$ and $\{\mathsf{IDEAL}^{V^*}(1^\lambda)\}_\lambda$ are computationally indistinguishable, where the random variable $\mathsf{REAL}^{V^*}(1^\lambda)$ is defined via the following experiment. In what follows we denote by $P_{\mathsf{ZK}1}$ the prover's algorithm in the first round, and similarly we denote by $P_{\mathsf{ZK}3}$ his algorithm in the third round.*

Experiment $\mathsf{REAL}^{V^*}(1^\lambda)$ *is defined as follows:*

1. *Run $P_{\mathsf{ZK}1}(1^\lambda, x, w; r)$ and obtain output rwi_1 to be sent to the verifier.*
2. *Run the verifier $V^*(1^\lambda, \mathsf{rwi}_1)$ and interpret its output as message rwi_2.*
3. *Run $P_{\mathsf{ZK}3}(1^\lambda, \mathsf{rwi}_2, x, w; r)$, where $P_{\mathsf{ZK}3}$ is the (honest) prover's algorithm for generating the third message of the WI protocol, and send its output rwi_3 to V^*.*
4. *Set a counter $i = 0$.*
5. *If $i < B$, then set $i = i + 1$, and V^* (given all the information so far) generates another message rwi_2^i, and receives the (honest) prover's message $P_{\mathsf{ZK}3}(\mathsf{rwi}_2^i, x, w; r)$. Repeat this step until $i = B$.*
6. *The output of the experiment is the view of V^*.*

Experiment $\mathsf{IDEAL}^{V^*}(1^\lambda)$ *is the output of the experiment* $\mathsf{Sim}^{V^*}(1^\lambda, x; r)$.

Imported Theorem 2. *[IKOS07] Assume the existence of injective one-way functions. Then, there exists a 3-round constant-soundness zero-knowledge protocol for proving NP statements, that is simulatable under B-bounded rewinding according to Definition 17.*

The work of [IKOS07] constructs such a protocol by having the prover emulate a $t = 2B$-private MPC "in his head".

Construction. We construct the protocol RWI with constant soundness error for the Graph Hamiltonicity problem, in Figure 7. Soundness of this protocol can be amplified via parallel repetition while preserving the WI property: and this can be used to prove general NP statements via a Karp reduction. The protocol RWI consists of 4 algorithms $(\mathsf{RWI}_1, \mathsf{RWI}_2, \mathsf{RWI}_3, \mathsf{RWI}_4)$ where the first 3 denote the algorithms used by the prover and verifier to send their messages and the last is the final verification algorithm.

We use the protocol from Imported Theorem 2. We denote its algorithms by $\mathsf{Head.ZK} = (\mathsf{Head.ZK}_1, \mathsf{Head.ZK}_2, \mathsf{Head.ZK}_3, \mathsf{Head.ZK}_4)$, where the first 3 denote the algorithms used by the prover and verifier to generate their messages, and the last is the final verification algorithm.

B 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

Inputs: Prover P obtains input (in the third round) an efficiently sampleable graph G (the statement) and a corresponding hamiltonian cycle H (the witness).

1. **Round 1: Prover message:**
   - Pick a random cyclic graph H', uniform randomness r and compute $\mathsf{nmcom} = \mathsf{Com}(H'; \mathsf{r})$ to the adjacency matrix of H' using a non-interactive commitment Com.
   - Compute $\mathsf{hzk}_1^{P \to V} \leftarrow \mathsf{Head.ZK}_1(1^\lambda)$ and send $(\mathsf{nmcom}, \mathsf{prove}_1^{P \to V})$ to V.
2. **Round 2: Verifier message:**
   - Pick a random bit b.
   - If $b = 0$, compute $\mathsf{hzk}_2^{V \to P} \leftarrow \mathsf{Head.ZK}_2(\mathsf{hzk}_1^{P \to V})$ and send $(0, \mathsf{hzk}_2^{V \to P})$ to P.
   - If $b = 1$, send $(1, \perp)$ to P.
3. **Round 3: Prover message:**
   - On input graph G with hamiltonian cycle H, do the following:
   - If $b = 0$, compute and send $\mathsf{hzk}_3^{P \to V} \leftarrow \mathsf{Head.ZK}_3(\mathsf{hzk}_1^{P \to V}, \mathsf{hzk}_2^{V \to P})$ to prove using witness $w = (H', \mathsf{r})$ that H' is a cyclic graph and $\mathsf{nmcom} = \mathsf{Com}(H'; \mathsf{r})$.
   - If $b = 1$, compute and send a permutation π that maps H' onto H. Also, decommit to all edges in the adjacency graph of H' that correspond to non-edges in $\pi^{-1}(G)$.
4. **Verifier Output:**
   - If $b = 0$, compute the output of the algorithm $\mathsf{Head.ZK}_4$.
   - If $b = 1$, check that all opened values are 0 and map (via π) to all non-edges of G.

Figure 7: 3 round Bounded Rewinding Secure WI

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 distinguisher-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 ECom:** In the main thread, for each honest party P_i, $\mathsf{Sim}_{\mathsf{Hyb}}$ does the following: in round 3, compute $\mathsf{ecom}_{b,3}^{i\to j} = \mathsf{ECom}_3(\mathsf{x}_i, \mathsf{r}_i, \mathsf{ecom}_{b,1}^{i\to j}, \mathsf{ecom}_{b,2}^{j\to i}; \mathsf{r}_{b,\mathsf{ecom}}^{i\to j})$. 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 first set of look-ahead threads. In the first set of look-ahead threads, the second extractable commitment is computed as in the main thread: that is, $\mathsf{ecom}_{b,3}^{i\to j} = \mathsf{ECom}_3(\mathsf{x}_i, \mathsf{r}_i, \mathsf{ecom}_{b,1}^{i\to j}, \mathsf{ecom}_{b,2}^{j\to i}; \mathsf{r}_{b,\mathsf{ecom}}^{i\to j})$. In the second set of look-aheads, the second extractable commitment continues to be computed using input 0: that is, $\mathsf{ecom}_{b,3}^{i\to j} = \mathsf{ECom}_3(0, \mathsf{r}_i, \mathsf{ecom}_{b,1}^{i\to j}, \mathsf{ecom}_{b,2}^{j\to i}; \mathsf{r}_{b,\mathsf{ecom}}^{i\to j})$ where r_i is sampled fresh in each look-ahead thread.

- H_3 - **Switch RWI:** In every thread (main and look-aheads), in round 3, $\mathsf{Sim}_{\mathsf{Hyb}}$ computes the RWI 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 RWI:** In the main thread, in round 3, $\mathsf{Sim}_{\mathsf{Hyb}}$ computes the RWI 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 ECom:** In the main thread, for each honest party P_i, $\mathsf{Sim}_{\mathsf{Hyb}}$ does the following: in round 3, compute $\mathsf{ecom}_{a,3}^{i\to j} = \mathsf{ECom}_3(0, \mathsf{r}_i, \mathsf{ecom}_{a,1}^{i\to j}, \mathsf{ecom}_{a,2}^{j\to i}; \mathsf{r}_{a,\mathsf{ecom}}^{i\to j})$. 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 RWI:** In every thread (main and look-aheads), in round 3, $\mathsf{Sim}_{\mathsf{Hyb}}$ computes the RWI 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 ECom:** In the main thread and every look-ahead thread in the first set, for each honest party P_i, $\mathsf{Sim}_{\mathsf{Hyb}}$ does the following: in round 3, compute $\mathsf{ecom}_{b,3}^{i\to j} = \mathsf{ECom}_3(\bot, \mathsf{ecom}_{b,1}^{i\to j}, \mathsf{ecom}_{b,2}^{j\to i}; \mathsf{r}_{b,\mathsf{ecom}}^{i\to j})$. That is, the simulator now commits to $\bot$ in the second extractable commitment scheme.

- 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 19.

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 s for the term $\mathsf{ecom}_{b,3,N+1}$ in each third round message. Informally, recall that this is the reusability property mentioned in Section 7.1. 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 $\mathsf{ecom}_{b,3}^{i\to j}$ as $(\mathsf{ecom}_{b,3,1}^{i\to j}, \ldots, \mathsf{ecom}_{b,3,N+2}^{i\to j})$ where $\mathsf{ecom}_{b,3,1}^{i\to j}, \ldots, \mathsf{ecom}_{b,3,N}^{i\to j}$ are the same in all the threads, $\mathsf{ecom}_{b,3,N+1} = s^{i\to j}$ is sampled afresh in each thread and $\mathsf{ecom}_{b,3,N+2} = \mathsf{PRF}(k^{i\to j}, s^{i\to j}) \oplus 0$. 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 non-interactive commitment* Com *and the pseudo-random function* PRF *used inside the scheme* ECom, H_1 *is computationally indistinguishable from* H_2.

Proof. Let's briefly recall the construction of the scheme ECom from Section 7.1 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 $\mathsf{ecom}_1^{i\to j} = \{\mathsf{ecom}_{1,1}^{i\to j}, \ldots, \mathsf{ecom}_{1,N}^{i\to j}\}$ where $\mathsf{ecom}_{1,\ell}^{i\to j} = \mathsf{Com}(\mathsf{p}_\ell^{i\to j})$ where each $\mathsf{p}_\ell^{i\to j}$ is a random polynomial of degree 4 (defined in Section 7.1). In round 2, P_j generates $\mathsf{ecom}_2^{j\to i} = (\mathsf{z}_1^{j\to i}, \ldots, \mathsf{z}_N^{j\to i})$ where each $\mathsf{z}_\ell^{j\to i}$ is a random value. Then, in round 3, P_i outputs $\mathsf{ecom}_3^{i\to j} = \{\mathsf{ecom}_{3,1}^{i\to j}, \ldots, \mathsf{ecom}_{3,N+2}^{i\to j}\}$ where $\mathsf{ecom}_{3,\ell}^{i\to j} = (k^{i\to j} \oplus \mathsf{p}_\ell^{i\to j}(0), \mathsf{p}_\ell^{i\to j}(\mathsf{z}_\ell^{j\to i}))$ for each $l \in [N]$ and $\mathsf{ecom}_{3,N+1} = s^{i\to j}$ and $\mathsf{ecom}_{3,N+2} = \mathsf{PRF}(k^{i\to j}, s^{i\to j}) \oplus m$.

We will now prove this subclaim via a series of intermediate sub-hybrids $\mathsf{Sub.Hyb}_1$ to $\mathsf{Sub.Hyb}_5$ where $\mathsf{Sub.Hyb}_1$ corresponds to H_1 and $\mathsf{Sub.Hyb}_5$ corresponds to H_2. Throughout, we skip the subscript "b" to ease the exposition.

- $\mathsf{Sub.Hyb}_1$**:** This is same as H_1.
- $\mathsf{Sub.Hyb}_2$**:** In the main thread and each look ahead thread in the first set, for every honest party P_i and malicious party P_j, in round 1, compute $\mathsf{ecom}_{1,\ell}^{i\to j} = \mathsf{Com}(0)$ for all $\ell \in [N]$.
  This is indistinguishable from the previous hybrid by the hiding property of the scheme Com.
- $\mathsf{Sub.Hyb}_3$**:** In the main thread and each look ahead thread in the first set, for every honest party P_i and malicious party P_j, pick $k_1^{i\to j}$ uniformly at random. Then, for each $\ell \in [N]$, pick a new degree B polynomial $\mathsf{q}_\ell^{i\to j}$ such that $(k^{i\to j} \oplus \mathsf{p}_\ell^{i\to j}(0)) = (k_1^{i\to j} \oplus \mathsf{q}_\ell^{i\to j}(0))$. Compute $\mathsf{ecom}_{3,\ell}$ as $(k_1^{i\to j} \oplus \mathsf{q}_\ell^{i\to j}(0), \mathsf{q}_\ell^{i\to j}(\mathsf{z}_\ell^{i\to j}))$.
  This hybrid is statistically indistinguishable from the previous hybrid.

- $\mathsf{Sub.Hyb}_4$**:** In the main thread and each look ahead thread in the first set, for every honest party P_i and malicious party P_j, in round 1, compute $\mathsf{ecom}^{i\to j}_{1,\ell} = \mathsf{Com}(\mathsf{q}^{i\to j}_{\ell})$ for all $\ell \in [N]$. This is indistinguishable from the previous hybrid by the hiding property of the scheme Com.

- $\mathsf{Sub.Hyb}_5$**:** In the main thread and each look ahead thread in the first set, for every honest party P_i and malicious party P_j, compute $\mathsf{ecom}^{i\to j}_{3,N+2} = \mathsf{PRF}(k^{i\to j}_1, \mathsf{ecom}^{i\to j}_{3,N+1}) \oplus \mathsf{x}_i$. 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 reusability security of the scheme* RWI*,* H_2 *is computationally indistinguishable from* H_3*.*

Proof. This follows from the reusability property of RWI in the same manner as in [JKKR17]. (In particular, we make L invocations of the reusability security, where L is the total number of threads.) □

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 reusability security of the scheme* RWI*,* H_4 *is computationally indistinguishable from* H_5*.*

Proof. The proof is 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 hiding property of the commitment scheme* Com *and the security of the pseudorandom function* PRF *used inside the scheme* 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 reusability 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 hiding property of the commitment scheme* Com *and the security of the pseudorandom function* PRF *used inside the scheme* 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 hiding property of the non-interactive commitment scheme* NCom*,* H_9 *is computationally indistinguishable from* H_{10}*.*

Proof. The proof is same as the proof of Sub-Claim 8 discussed above. □

Sub-Claim 15. *Assuming the Weak Zero Knowledge property of the argument system* WZK*,* H_{10} *is computationally indistinguishable from* H_{11}*.*

Proof. The proof is same as the proof of Sub-Claim 5 discussed above. □