Promise Zero Knowledge and its Applications to Round Optimal MPC

Saikrishna Badrinarayanan[1] *, Vipul Goyal[2] **, Abhishek Jain[3] ***, Yael Tauman Kalai[4], Dakshita Khurana[1] *, and Amit Sahai[1] *

[1] UCLA
{saikrishna,dakshita,sahai}@cs.ucla.edu
[2] CMU
goyal@cs.cmu.edu
[3] JHU
abhishek@cs.jhu.edu
[4] Microsoft Research, MIT
yael@microsoft.com

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 assuming polynomially hard DDH (or QR or N^{th}-Residuosity).

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 DDH (or QR or N^{th}-Residuosity).
- 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 DDH (or QR or N^{th}-Residuosity). This result generalizes to randomized input-less functionalities.

Previously, four round MPC protocols required sub-exponential-time hardness assumptions and no multi-party three-round protocols were

* Research supported in part from a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1619348, 1228984, 1136174, and 1065276, BSF grant 2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C-0205. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government. Fifth author's research also supported by the UCLA Dissertation Year Fellowship.

** Work supported in part by a grant from Northrop Grumman.

*** Supported in part by a DARPA/ARL Safeware Grant W911NF-15-C-0213, and a subaward from NSF CNS-1414023.

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 rely upon a *leveled rewinding security* technique that can be viewed as a polynomial-time alternative to leveled complexity leveraging for achieving "non-malleability" across different primitives.

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 [19], 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[5] is impossible with three or fewer rounds of simultaneous message exchange [17,14]. 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 [17,14]. 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 consider adversarial verifiers that promise not to abort the protocol early with noticeable probability. However,

[5] In this work, we focus on black-box simulation. However, no solutions for three-round ZK from standard assumptions with non-black-box simulation [2] are presently known either. [6] 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.

we do not limit ourselves only to adversarial verifiers that behave honestly; we 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 on DDH/QR/N^{th}-Residuosity assumption. 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 adversary should, generally speaking, only hurt the adversary, not help it. We know due to the impossibility results of [17,14], that we cannot leverage this insight to achieve three-round standard ZK (with black-box simulation). 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 [24].

Secure Multi-Party Computation. The notion of secure multiparty computation (MPC) [34,18] 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 [18,4,26,25,29,**?**,**?**,20,14,1,7,8,**?**]. 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. [1] and Brakerski et al. [7], 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 an unbiased coin (or a sequence of unbiased coins). Fair multiparty coin-tossing is known to be impossible in the dishonest majority setting [10]. 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.

Presently, secure multiparty coin-tossing is known to require at least four rounds w.r.t. black-box simulation [14,25]. In this work, we seek to overcome this barrier.

Towards this, our key observation is that coin-tossing is perfectly suited for application of partitioned simulation. The definition of secure coin-tossing roughly requires the existence of a simulator that successfully forces 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. Crucially, 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 the case of non-aborting adversaries, which is exactly the setting that promise ZK was designed for.

Using promise ZK, we design a three-round protocol for "list coin-tossing" – a notion that is slightly weaker that regular coin-tossing, but nevertheless, suffices for nearly all important applications of coin-tossing (see below for a discussion).

Therefore, promise ZK gives us a way to overcome the four-round barrier for secure coin-tossing [14,25].

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 DDH/QR/Nth-Residuosity.

Theorem 1 (Informal). *Assuming DDH/QR/Nth-Residuosity, 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 random strings 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 Nth-Residuosity. The resulting protocol is optimal in the number of rounds w.r.t. black-box simulation [14]. A three round semi-malicious protocol with the aforementioned property can be obtained based on DDH/QR/Nth Residuosity [15,5].

Theorem 2 (Informal). *Assuming DDH/QR/Nth-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 [14], 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 DDH/QR/Nth-Residuosity, there exists a three round multiparty list coin-tossing protocol with black-box simulation. This*

can be generalized to randomized inputless functionalities where security is defined analogously to list coin-tossing.

Finally, we note that by applying the transformation[6] of [14] 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 [25].

Corollary 1 (Informal). *Assuming DDH/QR/N^{th}-Residuosity, 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 [1,7], 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., [9]).

We instead rely upon a *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 remark that leveled rewinding security with a "single level" was previously used in [22]; here we extend this idea to "multiple levels".

1.2 Related Work

Concurrent Work. In a concurrent and independent work, Halevi et al. [23] construct a four round MPC protocol against malicious adversaries in the plain

[6] The work of Garg et al. [14] establishes an impossibility result for three round multi-party 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 [25] who proved the impossibility of four round two-party coin-tossing.

model based on different assumptions than ours. In particular, they rely upon enhanced trapdoor permutations and public-key encryption schemes that admit affine homomorphisms with equivocation (which in turn can be based on LWE/DDH/QR; see [23]). They do not consider the problems of promise ZK and list coin-tossing. We discuss more related work in the full version of the paper.

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.[7] 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 DDH/QR/Nth-Residuosity. We work in the delayed-input setting where the statement being proven is revealed to the (adversarial) verifier only in the last round.[8] Further, we work in the distributional setting, where statements being proven are sampled from an efficiently sampleable public distribution, i.e., it is possible to efficiently sample a statement together with a witness.

For simplicity of presentation, here we describe our construction using an additional assumption of two-round WI proofs, a.k.a. Zaps [12]. 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 Assuming DDH/QR/Nth-Residuosity.[9]

Our construction of promise ZK roughly follows the FLS paradigm [13] 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.

[7] 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.

[8] 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.

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

- 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.

To address this, we observe 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). This is an example of leveled rewinding security technique with a single level; later we discuss its application with multiple levels.

We note that standard extractable commitment schemes such as [31,32] (as well as their delayed-input variants) 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[10] 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.

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

Now, since the adversarial verifier may be rushing, the simulator must first provide its third round message (namely, the second message of Zap) on each look-ahead thread in order to learn the verifier's third round message. Since the simulator does not have a trapdoor yet, the only possibility for the simulator to prepare a valid third round message is by behaving honestly. However, the 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).[11]

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 public coin) 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 several previous works, our starting idea is to follow the GMW paradigm [18] 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 [17,14] and even with non-black box simulation, are not known from standard assumptions.

Promise ZK and Partitioned Simulation. While [1,7] 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

[11] The idea of using a witness to continue simulation is an old one [3]. Most recently, [24] used this idea in the distributional setting.

is aborting in the third round. For now, let us focus on the case where the adversary does not abort in the third round of Σ; later we 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).[12] We also remark that in our actual construction, to address non-malleability concerns [11], the promise ZK and the standard ZK protocols that we use are suitably "hardened" using three-round non-malleable commitments [21,27] to achieve *simulation soundness* [33] 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.

We now proceed to discuss the main technical challenges underlying our construction and its proof of security.

How to do "Non-Malleable" Input Extraction? Let us start with the issue of extraction of adversary's input and trapdoors (for simulation of ZK proofs). In the aforementioned protocol design, 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.[13]

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

[12] 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.

[13] Indeed, [1] implement such a strategy in their security proof by relying on sub-exponential hardness assumptions.

extractable. Thus, it may seem that we have reached an impasse.

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

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

Using Promise ZK. In the works of [1,7], the simulator behaves honestly in the first three rounds using random inputs for the honest parties. *We depart from this proof strategy, and instead, require our simulator to cheat even in the first three*

rounds on the main thread.[14] Indeed, such a simulation strategy seems necessary for our case since the recent two-round semi-malicious MPC protocols of [15,5] – which we use to instantiate our compiler – require a cheating simulation strategy even in the first round.

To implement this proof strategy, we turn to promise ZK. However, recall that promise ZK simulator works by behaving honestly in the look-ahead threads. When applied to our MPC construction, this means that we must find a way to behave honestly on the look-ahead threads that are used for extracting inputs and trapdoors from the adversary. However, at first it is not immediately clear how to implement such a strategy. Clearly, our final simulator cannot use honest party inputs on the look-ahead threads to behave honestly.

Instead, our simulator uses random inputs to behave honestly on the look-ahead threads. The main challenge then is to argue that when we switch from using real honest inputs (in an intermediate hybrid) to random inputs on the look-ahead threads, the probability of extraction of adversary's inputs and trapdoors remains unchanged. Crucially, here, we do not need to consider a joint view across all the look-ahead threads, and instead, it suffices to argue the indistinguishability of adversary's view on each look-ahead thread (when we switch from real input to random input) one at a time. We rely upon techniques from the work of Jain et al. [24] for this indistinguishability argument. The same proof technique is also used to argue security in the case when the adversary aborts in the third round with overwhelming probability. We discuss this next.

Aborting Adversary Case. In the case where the adversary aborts in the third round with overwhelming probability, 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). Therefore, in this case, the simulator simply behave honestly on the main thread using random inputs (as in [1,7]). The main challenge then is to switch in an indistinguishable manner from honest behavior in the first three rounds using real inputs to honest behavior using random inputs, while relying only on polynomial-time assumptions.

We address this case by relying upon techniques from [24]. We remark that we cannot directly use the three-round strong WI argument system of [24] since it requires the instance being proven to be disclosed to the verifier only in the third round of the protocol. This is not true in our case, since the instance also consists of the transcript of the three-round extractable commitment (and other sub-protocols like the trapdoor generation). Nevertheless, we are able to use ideas from [24] in a non-black-box manner to enable our security proof; we refer the reader to the technical sections for more details.

Other Issues. We note that since our partitioned simulation technique crucially relies upon identifying whether an adversary is aborting or not, we have to take

[14] We emphasize that this strategy is only used in the case where the adversary does not abort in the third round. As we discuss below, we use a different strategy in the aborting adversary case.

precaution during simulation to avoid the possibility of the simulator running in exponential time. For this reason, we use ideas first developed in [17] and later used in many subsequent works, to ensure that the running time of our simulator is expected polynomial-time.

Finally, we note that the above discussion is oversimplified, and omits several technical points. We refer the reader to the technical sections for full details.

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 DDH/QR/Nth-Residuosity 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 may seem that we have successfully constructed a three round multiparty coin-tossing protocol, which would contradict the lower bound of [14]!

We now explain the flaw in the above argument. As is typical to security with abort, 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 define and construct Simulation-Extractable Promise ZK. Due to lack of space, our three round List Coin Tossing protocol and our four round maliciously secure MPC protocol are described in the full version of the paper.

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 [16] 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

Here onwards until Section 5, we will discuss protocols where only one party sends a message in any round.

Definition 2 (Extractable Commitments). *Consider any statistically binding, computationally hiding commitment scheme $\langle C, R \rangle$. Let $\mathsf{Trans}\langle C(m, r_C), R(r_R)\rangle$ denote a commitment transcript with committer input m, committer randomness r_C and receiver randomness r_R, and let $\mathsf{Decom}(\tau, m, r_C)$ denote the algorithm that on input a commitment transcript τ, committer message m and randomness r_C outputs 1 or 0 to denote whether or not the decommitment was accepted (we explicitly require the decommitment phase to not require receiver randomness r_R).*

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 $\mathsf{Trans}\langle C^*, R(r_R)\rangle$ *denote a transcript of the interaction between* C^* *and* R*. Then* $\mathcal{E}^{C^*}(\mathsf{Trans}\langle C^*, R(r_R)\rangle)$ *outputs* m, r_C *such that over the randomness of* $\mathcal{E}$ *and of sampling* $\mathsf{Trans}\langle C^*, R(r_R)\rangle$*:*

$$\Pr[(\exists \widetilde{m} \neq m, \widetilde{r}_C) \text{ such that } \mathsf{Decom}(\tau, \widetilde{m}, \widetilde{r}_C) = 1] = \mathsf{negl}(\lambda)$$

Remark 2. The notion of extraction described in Definition 2 is often referred to as over-extraction. This is because the extractor $\mathcal{E}$ is allowed to output any arbitrary value if $\mathsf{Trans}\langle C^*, R(r_R)\rangle$ does not contain a commitment to any valid message. On the other hand, if $\mathsf{Trans}\langle C^*, R(r_R)\rangle$ is a valid commitment to some message m, $\mathcal{E}$ must output the correct committed message m.

Definition 3 (k-Extractable Commitments). *An extractable commitment satisfying Definition 2 is said to be* k*-extractable if there exists a polynomial* $p(\cdot)$ *such that the extractor* $\mathcal{E}^{C^*}(\mathsf{Trans}\langle C^*, R(r_R)\rangle)$ with $k-1$ queries to C^**, outputs* m, r_C *such that over the randomness of* $\mathcal{E}$ *and of sampling* $\mathsf{Trans}\langle C^*, R(r_R)\rangle$*:*

$$\Pr[\mathsf{Decom}(\tau, m, r_C) = 1] \geq p(\lambda)$$

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.

Theorem 4. *[31,32] For any constant* $K > 0$*, assuming injective one-way functions, there exists a three round delayed-input* K*-extractable commitment scheme satisfying Definition 3.*

3.4 Non-Malleable Commitments

We start with the definition of non-malleable commitments by Pass and Rosen [30] and further refined by Lin et al [28] and Goyal [20]. (All of these definitions build upon the original definition of Dwork et al. [11]).

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 val and MIM receives an auxiliary input z, which might contain a priori information about val. 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[15] *PPT* MIM*, there exists a PPT simulator* $\mathcal{S}$ *such that the following ensembles are computationally indistinguishable:*

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

- $\langle C, R\rangle$ *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 a message* com_2*.*
  3. *If* $b = 0$*, run* $C(\sigma, \mathsf{com}_2)$ *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. [21] construct three-round special non-malleable commitments satisfying Definition 4 based on injective OWFs.

Imported Theorem 1 ([21]) *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.

[15] 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 [31,32,8]. 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 [8].

Syntax. A trapdoor generation protocol

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

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

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

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

Extraction. There exists a PPT extractor algorithm TDExt that, given a set of values[16] $(\mathsf{td}_1, \{\mathsf{td}_2^i, \mathsf{td}_3^i\}_{i=1}^3)$ such that $\mathsf{td}_2^1, \mathsf{td}_2^2, \mathsf{td}_2^3$ are distinct and $\mathsf{TDOut}(\mathsf{td}_1, \mathsf{td}_2^i, \mathsf{td}_3^i) = 1$ for all $i \in [3]$, outputs a trapdoor t such that $\mathsf{TDValid}(\mathsf{t}, \mathsf{td}_1) = 1$.

1-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.*

Construction We describe our construction of a three round trapdoor generation protocol based on one way functions in the full version of the paper.

4.2 WI with Non-adaptive 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 polynomially hard DDH (or QR or N^{th}-Residuosity). In the non-delayed-input setting, such a primitive was implicitly constructed and used previously by Goyal et al. [22].[17]

We formally define three-round delayed-input WI with *non-adaptive* bounded-rewinding security here. In the full version, we describe a construction for the same. For our applications, we instantiate the rewinding parameter B with the value 6.

[16] 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.

[17] Specifically, they consider non-delayed-input WI with 1-rewinding security.

Definition 6 (3-Round Delayed-Input WI with Non-adaptive 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* Non-adaptive 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 $\{(x^i, w^i)\}_{i\in[B-1]}$, x^B, w_0^B, w_1^B and a set of messages $\{\mathsf{rwi}_2^i\}_{i\in[B]}$.*
3. *For each $i \in [B-1]$, run $P_3(\sigma, \mathsf{rwi}_2^i, x^i, w^i)$, and for $i = B$, run $P_3(\sigma, \mathsf{rwi}_2^i, x^i, w_b^i)$ where P_3 is the (honest) prover's algorithm for generating the third message of the WI protocol, and send its message $\{\mathsf{rwi}_3^i\}_{i\in[B]}$ to V^*.*

In the full version, we prove the following theorem:

Theorem 5. *Assuming DDH/QR/N^{th}-Residuosity, there exists a three round delayed-input witness-indistinguishable argument system with non-adaptive ($B = 6$)-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 (that do not send invalid messages).

Validity Check. First, 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 the prover does not need to explicitly respond to this message[18] 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 if the verifier's messages are generated honestly according to the protocol, then

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

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.

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. Second, 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 V^* conditioned on $\mathsf{view}_{V^*,1}$.

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

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

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.

[18] 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 **??**.

Definition 7. *For any constant $c \in \mathbb{N}$, 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 $c \in \mathbb{N}$, 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 $\mathcal{D} = \{\mathcal{D}_\lambda\}$ 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 (defined below) hold for REAL_{V^*} and IDEAL_{V^*} (defined below).*

- *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 $(\mathsf{view}_{V^*,1}, \langle P_2(x, w, \mathsf{st}), V^*(\mathsf{view}_{V^*,1})\rangle)$, where V^* obtains x_i in round i.*
  - *The output of the experiment is the view of V^* in the execution $(x, \langle P(x, w), V^*(\mathsf{view}_{V^*,1})\rangle)$.*
- *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 $(\mathsf{view}_{V^*,1}, \langle P_2(x, w, \mathsf{st}), V^*(\mathsf{view}_{V^*,1})\rangle)$, where V^* obtains x_i in round i.*
    - *The output of the experiment is $(x, \langle P(x, w), V^*(\mathsf{view}_{V^*,1})\rangle)$.*
  - *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:*
    - *Compute $\mathsf{Sim}_1^{V^*}(\mathsf{view}_{V^*,1}, \mathsf{st}, p) \to \mathsf{st}_1$.*
    - *Then 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 $(\mathsf{view}_{V^*,3})$.*

Conditions 1 and 2 are defined 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* $\mathsf{Sim}_2^{V^*}(x_2, \mathsf{view}_{V^*,1}, \mathsf{st})$ *and* $\mathsf{Sim}_3^{V^*}(x_3, \mathsf{view}_{V^*,1}, \mathsf{view}_{V^*,2}, \mathsf{st}_2)$ *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 the 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 the 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 $\mathsf{Trans}_{\text{left}}(P_2(x, w, \mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}), \mathsf{MIM})$ 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: $\mathsf{Trans}_{\text{left}}(P, \mathsf{MIM}) = (\mathsf{view}_{\mathsf{MIM},1}, \mathsf{Trans}_{\text{left}}(P_2(x, w, \mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}), \mathsf{MIM}))$. For any $\mathsf{view}_{\mathsf{MIM},1}$ sampled according to honest prover and verifier strategy as described above, let $q_{\mathsf{view}_{\mathsf{MIM},1}} =$

$$\Pr[\mathsf{Valid}(\mathsf{view}_{\mathsf{MIM},1}, \mathsf{Trans}_{\text{left}}(P_2(x, w, \mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}), \mathsf{MIM})) = 1 | (\mathsf{view}_{\mathsf{MIM},1}, \mathsf{st}) \leftarrow P_1(1^\lambda)]$$

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. *For any constant $c \in \mathbb{N}$, 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 $\frac{p}{2} < q_{\mathsf{view}_{\mathsf{MIM},1}} < 2 \cdot p$.*

Remark 3. We briefly describe a canonical polynomial-time validity approximation algorithm for any constant $c \in \mathbb{N}$:

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 $c \in \mathbb{N}$, 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 (both left and right executions) 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*[19] *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*,*[20] *where* $\mathsf{view}_{\mathsf{MIM},1}$ *denotes all the simulated messages and* st *denotes the secret states of all the honest parties, and*

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

[19] 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.

[20] 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* $\mathsf{Valid}(\mathsf{Trans}_{\mathrm{left}}(P_2(x, w, \mathsf{st})))$ *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 6. *Assuming the existence of polynomially hard DDH/QR/*N^{th}*-Residuosity, there exists a three round simulation-sound promise ZK argument according to Definition 10.*

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 non-adaptive bounded rewinding security for $B = 6$ 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 :

$$\Bigg(R(x,w) = 1\Bigg) \bigvee \Bigg(\mathsf{TDValid}(\mathsf{td}_1, \mathsf{t}) = 1 \wedge \mathsf{c}_1 = \mathsf{NMCom}_1(\mathsf{r}) \wedge \mathsf{c}_3 = \mathsf{NMCom}_3(\mathsf{t}, \mathsf{c}_1, \mathsf{c}_2; \mathsf{r})\Bigg).$$ 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 1. The completeness of this protocol follows from the correctness of the underlying primitives.

5.3 Security Proof

Due to lack of space, we defer the proof of our protocol to the full version.

Acknowledgements. We thank Silas Richelson, Shai Halevi, Carmit Hazay, Antigoni Polychroniadou, Muthuramakrishnan Venkitasubramaniam and the anonymous reviewers of STOC 2018 for useful comments in an earlier draft of this paper.

References

1. Ananth, P., Choudhuri, A.R., Jain, A.: A new approach to round-optimal secure multiparty computation. In: CRYPTO. pp. 468–499 (2017)
2. Barak, B.: How to go beyond the black-box simulation barrier. In: Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on. pp. 106–115. IEEE (2001)
3. Barak, B., Sahai, A.: 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. pp. 543–552 (2005), https://doi.org/10.1109/SFCS.2005.43
4. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: STOC. pp. 503–513 (1990)
5. Benhamouda, F., Lin, H.: k-round mpc from k-round ot via garbled interactive circuits. EUROCRYPT (2018)
6. Bitansky, N., Kalai, Y.T., Paneth, O.: Multi-collision resistance: A paradigm for keyless hash functions. IACR Cryptology ePrint Archive 2017, 488 (2017), http://eprint.iacr.org/2017/488

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.

Fig. 1: Three round Simulation-Sound Promise ZK argument.

7. Brakerski, Z., Halevi, S., Polychroniadou, A.: Four round secure computation without setup. In: TCC (2017)
8. Ciampi, M., Ostrovsky, R., Siniscalchi, Visconti, I.: Delayed-input non-malleable zero knowledge and multi-party coin tossing in four rounds. In: TCC (2017)
9. Ciampi, M., Ostrovsky, R., Siniscalchi, L., Visconti, I.: Concurrent non-malleable commitments (and more) in 3 rounds. In: CRYPTO. pp. 270–299 (2016)
10. Cleve, R.: Limits on the security of coin flips when half the processors are faulty (extended abstract). In: Hartmanis, J. (ed.) STOC. pp. 364–369. ACM (1986)
11. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). In: STOC. pp. 542–552 (1991)
12. Dwork, C., Naor, M.: Zaps and their applications. In: FOCS. pp. 283–293 (2000)

13. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In: FOCS. pp. 308–317 (1990)
14. Garg, S., Mukherjee, P., Pandey, O., Polychroniadou, A.: The exact round complexity of secure computation. In: EUROCRYPT. pp. 448–476 (2016)
15. Garg, S., Srinivasan, A.: Two-round multiparty secure computation from minimal assumptions. EUROCRYPT (2018)
16. Goldreich, O.: The Foundations of Cryptography - Volume 2, Basic Applications. Cambridge University Press (2004)
17. Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. J. Cryptology 9(3), 167–190 (1996)
18. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: STOC. pp. 218–229 (1987)
19. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. (1989)
20. Goyal, V.: Constant round non-malleable protocols using one way functions. In: STOC. pp. 695–704 (2011)
21. Goyal, V., Pandey, O., Richelson, S.: Textbook non-malleable commitments. In: STOC. pp. 1128–1141 (2016)
22. Goyal, V., Richelson, S., Rosen, A., Vald, M.: An algebraic approach to non-malleability. In: FOCS. pp. 41–50 (2014)
23. Halevi, S., Hazay, C., Polychroniadou, A., Venkitasubramaniam, M.: Round-optimal secure multi-party computation. IACR Cryptology ePrint Archive. 2017, 1056 (2017), http://eprint.iacr.org/2017/1056, accepted to CRYPTO 2018.
24. Jain, A., Kalai, Y.T., Khurana, D., Rothblum, R.: Distinguisher-dependent simulation in two rounds and its applications. In: CRYPTO (2017)
25. Katz, J., Ostrovsky, R.: Round-optimal secure two-party computation. In: CRYPTO. pp. 335–354 (2004)
26. Katz, J., Ostrovsky, R., Smith, A.D.: Round efficiency of multi-party computation with a dishonest majority. In: EUROCRYPT. pp. 578–595 (2003)
27. Khurana, D.: Round optimal concurrent non-malleability from polynomial hardness. In: Kalai, Y., Reyzin, L. (eds.) Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10678, pp. 139–171. Springer (2017), https://doi.org/10.1007/978-3-319-70503-3_5
28. Lin, H., Pass, R., Venkitasubramaniam, M.: Concurrent non-malleable commitments from any one-way function. In: TCC. pp. 571–588 (2008)
29. Pass, R.: 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. pp. 232–241 (2004)
30. Pass, R., Rosen, A.: Concurrent non-malleable commitments. In: FOCS. pp. 563–572 (2005)
31. Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: FOCS. pp. 366–375 (2002)
32. Rosen, A.: A note on constant-round zero-knowledge proofs for NP. In: TCC. pp. 191–202 (2004)
33. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: FOCS. pp. 543–553 (1999)
34. Yao, A.C.: Protocols for secure computations (extended abstract). In: FOCS (1982)