Non-Malleable Zero Knowledge: Black-Box Constructions and Definitional Relationships

Abhishek Jain* Omkant Pandey†

Abstract

This paper deals with efficient non-malleable zero-knowledge proofs for $\mathcal{NP}$, based on general assumptions. We construct a simulation-sound zero-knowledge ($\mathcal{ZK}$) protocol for $\mathcal{NP}$, based only on the *black-box* use of *one-way functions*. Constructing such a proof system has been an open question ever since the original work of Dolev, Dwork, and Naor [DDN91]. In addition to the feasibility result, our protocol has a *constant* number of rounds, which is asymptotically optimal.

Traditionally, the term non-malleable zero-knowledge (NmZK) refers to the original definition of [DDN91]; but today it is used loosely to also refer to *simulation-soundness* (SimSound) [Sah99], and *simulation-extractability* (SimExt) [PR05b]. While SimExt implies NmZK, the common perception is that SimExt is strongest of the three notions. A formal study of the definitional relationship between these three notions, however, has never been done.

In the second part of this work, we try to correct this situation by initiating such a study. We show that in the "static" case, if an NmZK protocol is also an *argument-of-knowledge*, then it is in fact SimExt. Furthermore, in the most strict sense of the definition, SimSound does not necessarily follow from SimExt. These results are somewhat surprising because they are opposite to the common perception that SimExt is the strongest of the three notions.

1 Introduction

The concept of non-malleability was introduced in the seminal work of Dolev, Dwork, and Naor [DDN91, DDN00]. It has proven fundamental to several developments in cryptography such as CCA2-secure encryption schemes [NY90, DDN91, Sah99], composable multiparty computation [CLOS02, Pas04, LPV09], privacy amplification and non-malleable extractors [DW09], tamper-resilience [DPW10], hash functions [BCFW09], etc.

This paper is about non-malleability in zero-knowledge ($\mathcal{ZK}$) interactive proofs [GMR85]. Roughly speaking, [DDN91] define a $\mathcal{ZK}$ proof to be non-malleable, if no man-in-the-middle can improve his chances in proving a statement $\tilde{x}$ *even if* it receives a proof for a "related" statement x. An $O(\log n)$ round NmZK protocol for $\mathcal{NP}$, was given in [DDN91] based on the existence of one-way functions (Owf). The protocol of [DDN91] makes *non-black-box* use of one-way functions since it uses Cook-Levin reductions to $\mathcal{NP}$-complete problems [Coo71, Kar72, Lev84]. Such reductions are usually a large source of inefficiency both in terms of computation and communication, and are highly undesirable.

*UCLA. E-mail: abhishek@cs.ucla.edu.

†Microsoft Redmond. Email:omkantp@microsoft.com.

Black-box Constructions. A protocol A is said to make only *black-box* use of a cryptographic primitive B if the implementation of A uses B only as an *oracle* —i.e., only refers to the input/output behavior of B. Such a black-box construction does not depend on the actual implementation details of B, and therefore precludes the use of Cook-Levin reductions in the protocol explicitly. This results in much more efficient protocols, which are more suitable for actual implementations.

Despite intensive research on non-malleable $\mathcal{ZK}$ in recent years [DDN91, Sah99, DDO+01, Bar02, Pas04, PR05b, PR05a, PPV08, LP09, PW10, Wee10, Goy11, LP11], the complexity of NmZK protocols from the point of view of black-box use of cryptographic assumptions (such as Owf), is still not very well understood. And in particular, the following question has remained open thus far:

Do non-malleable $\mathcal{ZK}$ proofs, based only on black-box *use of general assumptions, exist?*

This stands in sharp contrast to the case of *non-malleable commitment* schemes, where we know much better. The following works show that non-malleable commitment schemes based only on black-box use of Owfs exist: Pass and Wee [PW09], Wee [Wee10], and finally Goyal [Goy11] in $O(\log n)$, $O(\log^* n)$, and $O(1)$ rounds, respectively.[1]

Round Complexity. Traditionally, in theory of cryptography, the round-complexity of protocols has often been an important measure of efficiency. Recent implementations show that the latency of sending and receiving messages back and forth can in fact be a dominating factor in running cryptographic protocols [MNPS04, BNP08]. Unlike the "black-box use" complexity, however, the round-complexity of non-malleable $\mathcal{ZK}$ is much well understood. In a series of results, round-efficient non-malleable protocols for both, $\mathcal{ZK}$ and commitments, based on general assumptions were presented in [Bar02, PR05b, LP09, PW10, Wee10]. Very recently and independently, Goyal [Goy11] and Lin-Pass [LP11] have constructed *constant round* non-malleable $\mathcal{ZK}$ protocols for all of $\mathcal{NP}$, assuming only one-way functions. Both of these works, however, still use the Owf in a non-black-box manner. In light of these recent results, we can make our previous question even more strict:

Do constant-round *non-malleable $\mathcal{ZK}$ proofs, based only on* black-box *use of general assumptions, exist?*

The first part of this work resolves this question in the affirmative. Specifically, we construct a *simulation-sound* $\mathcal{ZK}$ protocol for $\mathcal{NP}$, based on the existence of a *non-malleable commitment* scheme. In addition, the protocol uses this commitment scheme only as a black-box and adds only a constant number of rounds more. Then, by instantiating our protocol with a (suitable) non-malleable commitment schemes we get the desired result. In particular, by using [PW09], we get an $O(\log n)$ round solution, and using Goyal [Goy11] gives a constant round solution. We note that using Goyal's scheme [Goy11] is in fact more difficult since it does not satisfy the standard notion of "non-malleability w.r.t. commitment" [PR05a]. Instead, it satisfies the so called notion of "non-malleability w.r.t. replacement" in the one-many setting [PR05b, PR05a]. Nevertheless, we are able to argue security of our protocol even in this case.

[1] The black-box constructions of Wee [Wee10] and Goyal [Goy11] only satisfy slightly weaker notions of non-malleability: namely, "non-malleability w.r.t. extraction" and "non-malleability w.r.t. replacement" respectively.

Relationship between Various Notions of Non-malleability. As noted earlier, today the term non-malleable $\mathcal{ZK}$ is used loosely and can refer to any of the following three notions: NmZK as formulated by DDN [DDN00] (see also [PR05b]), simulation soundness (SimSound) as formulated by Sahai [Sah99] in the context of non-interactive zero-knowledge [BFM88, BSMP91],[2], and simulation-extractability (SimExt) as formulated by Pass and Rosen [PR05b] (see also [DDO+01, Lin01]). Briefly, NmZK requires that for every man-in-the-middle M, there exists an M^* who can succeed in proving the concerned statement $\tilde{x}$ on its own with almost the same probability (in a given $\mathcal{ZK}$ protocol). SimSound requires that no M can prove a false statement *even when* it can receive proofs to many false statements from the simulator. Finally, SimExt requires that there exist a *single* machine called the "simulator-extractor" which simulates the view of M and output witnesses for all statements that M proves successfully.

We note that while SimExt is *perceived* as the strongest notion of all three, an exact relationship between the three notions of non-malleability (as described above) is not known. In particular, the only known result in this context is due to Pass and Rosen [PR05b] who proved that SimExt implies NmZK. Given the importance of non-malleable $\mathcal{ZK}$ in cryptography, this is not a good state of affairs. Since definitions represent our intuitive understanding of a given cryptographic object, various definitions should be consistent in representing this object. Knowing the relationship between the various definitions tells us what exactly can be expected from a given security notion. Indeed, studying the formal relationships between various security notions is a well established line of research. For example, see the works in [BDPR98, KY00, BFOR08, BFO08] for an extensive study of such relationships between security notions of encryption schemes.

In the second part of this work, we take this direction and consider the *definitional equivalence* of these three notions. If a protocol satisfies a security notion A, does it also satisfy some other security notion B? If yes, then we write $A \Rightarrow B$, otherwise we write $A \not\Rightarrow B$. Notation $A \Leftrightarrow B$ implies that the two notions are equivalent, i.e. $A \Rightarrow B$, and $B \Rightarrow A$. Barring some technicalities, we arrive at the following conclusions:

$$\text{NmZK-AoK} \Leftrightarrow \text{SimExt} \not\Rightarrow \text{SimSound}$$

Here NmZK-AoK is short for NmZK argument-of-knowledge. Pass and Rosen [PR05b] showed that SimExt implies NmZK-AoK, and the common perception is that it is in fact stronger than the other two notions. Our conclusion are therefore somewhat surprising.

Other Related Works. In this paper, we only focus on constructions based on general assumptions, and in the plain model. If one is willing to give up on general assumptions, then under specific number-theoretic assumptions, efficient constructions are known for both non-malleable zero knowledge and commitments, e.g., see [PPV08, OPV10]. Likewise, if one is willing to depart from the plain model, and use setup assumptions such as a CRS, then efficient constructions are known for both non-malleable zero knowledge and commitments, e.g. see [CKOS01, GMY06, Lin11].

1.1 Overview of Main Ideas

The Simulation Sound Protocol. Intuitively, to construct NmZK, we should use a non-malleable commitment (NmCom) to obtain some sort of "independence" between left and right

[2]We note that simulation-soundness *in the interactive case* has never been explicitly formalized in the plain model; Garay, MacKenzie, and Yang presented a formulation for the same in the CRS model [GMY06]. We present a formulation in the plain model by building upon [Sah99, GMY06, PR05b].

execution. Consider using NmCom in Blum's Hamiltonicity (BH) protocol. Clearly, replacing the commitment scheme in BH by NmCom will not work since NmCom is not necessarily *concurrent non-malleable* [DDN91, PR05a].[3] Another idea is to use a Feige-Shamir style construction [FS89, FS90] to establish a "trapdoor" and commit to a fake witness using NmCom. Clearly, this approach does not suffer from the earlier issue of concurrent non-malleability. However, any such approach where NmCom will not be later "opened" must somehow prove something about the committed value (since otherwise, intuitively, such a commitment is of no use in the protocol). Note that this will require one to use Cook-Levin reductions unless one is able to leverage some "cut-and-choose" style techniques. Unfortunately, however, there is no hope of using "cut-and-choose" style protocols to circumvent this issue, since such protocols will run into the issues of concurrent non-malleability.

With these observations, our initial idea is to change the direction of the use of NmCom, i.e., construct a protocol where the verifier (instead of prover) commits to some value using NmCom, which must be opened later. One possible approach is to have the prover P and the verifier V perform coin-tossing to determine the "challenge" in the BH-protocol in the following manner: V first commits to a random string r_1 using NmCom and then opens it after P sends a random string r_2; the BH-challenge is then set to be $r_1 \oplus r_2$.

While a-priori this sounds like a promising approach, unfortunately, we run into the following problem in the proof. Note that during the simulation, the simulator S must simulate the outcome of the above coin-tossing phase to a fixed challenge value (say) ch for which it knows how to respond successfully. Therefore, in the simulated experiment, there will be sufficient information about ch in the first message of BH. This can be exploited by the man-in-the-middle M while preparing its NmCom, thus failing the simulation.

To this end, our next idea is to move the verifier's commitment to the "top" of the protocol. Specifically, we adopt the Goldreich-Kahan [GK96] approach and require V to commit its challenge ch using NmCom *before it sees any information* from P (or S) — use of NmCom guarantees that M's challenge will be independent of V's. Since NmCom used will be statistically binding[4], this approach compromises even the *stand alone* soundness of the scheme. Our *first key-idea* is to fix this stand-alone soundness. We do this by noting that the BH protocol has a special structure: once prover sends its message, then *for a false statement* x', there is at most *one* challenge ch' for which he can succeed. Therefore, to succeed, it must set $ch' = ch$. Hence, we replace the commitment scheme in BH-protocol by a special 3-round extractable commitment scheme [PRS02, Ros04]; this ensures that if prover sets $ch' = ch$, we can extract ch' using the extractable-commitment, without ever opening it.

When we move to the non-malleability setting, however, this approach runs into a second problem. To simulate a false proof, our simulator S will indeed rewind M and prepare the first message of BH, based on the value revealed by M, say ch_M. It is possible that M will set its value in right interaction, depending upon the message on its left. To overcome this situation, our *second key-observation* is to argue independence *without rewinding* M *at all*. In particular, we use the strong definition of non-malleability, which simply gives the value ch_M to S (or the distinguisher in the non-malleability experiment—see, e.g., [LPV08, PR05b]). Therefore, using this we can show

[3] Indeed there is no known construction of concurrent non-malleable commitments for tags of length n that uses Owf as a black-box. Furthermore, even if there existed one, it may not be suffice here since some of the commitments sent by the prover are opened later in the third round of BH protocol.

[4] In case of statistical hiding, we must rely on "non-malleability w.r.t. opening", and that is a troublesome proposition since no single value is defined in the commitment. This creates several new difficulties of its own.

that if M's preparation of the first BH-message on right changes as we go from real to the simulated world, we can once again extract using our first idea, while simulating in straight-line on left. A non-synchronous adversary is handled using simple scheduling arguments and standard-techniques.

This essentially summarizes the high-level ideas in our protocol. To obtain a constant round construction, we need to work a little more since the only known black-box scheme of Goyal [Goy11] satisfies only a weaker notion of non-malleability then what the above approach needs.

Definitional Relationships. We briefly highlight here how we obtain these results. First, it is not hard to see that NmZK-AoK implies SimExt if non-malleability property uses the adversary only as a *black-box*. That is, if M^* (guaranteed by NmZK) uses M as a black-box only, then M^* and the AOK property can be (easily) combined to construct a "simulator-extractor" for SimExt property. However, in general, it is not obvious if such a simulator-extractor can be constructed at all.

We show that in the *static* case, where M declares $\tilde{x}$ (after seeing x) before any execution begins, we can in fact construct such a simulator-extractor by combining M^* with the knowledge extractor E guaranteed by AOK property. The central difficulty that arises is that the success probability of M^* may differ negligibly from that of M in the real world. Since the extraction of the witness, intuitively, must come from M^* via rewinding, this negligible difference can result in exponential time during extraction. This is a problem akin to Goldreich-Kahan [GK96], and we use their techniques to overcome this situation. To the best of our knowledge, this is the first time that the techniques of [GK96] (to ensure expected polynomial-time simulation) have been applied in the context of non-malleability. The ideal result, of course, would be to prove equivalence even for the adaptive case. We leave this as an important open question.

Finally, the result SimExt $\not\Rightarrow$ SimSound is obtained by exploiting the fact that SimExt does not require any security guarantee in the case when the input message is false. Since SimSound *explicitly* deals with false messages, we are able to exploit this situation to obtain a counter-example.

2 Basic Definitions

We assume familiarity with standard definitions such as commitment schemes, interactive proofs, zero knowledge, and arguments of knowledge. For completeness, some of these notions in Appendix B.2. In this section, we first describe the man-in-the-middle setting for commitment protocols and then move on to the case of interactive proofs. Let $\langle C, R\rangle$ be a commitment protocol where C is the algorithm for honest committer and R is the algorithm for honest receiver. The protocol may be interactive, i.e., may require C to send more than one message to R. We assume that $\langle C, R\rangle$ is a string commitment scheme for strings at least as long as the security parameter, say n. Also, we assume that $\langle C, R\rangle$ is statistically binding (and hence only computationally hiding).

Man-in-the-Middle Experiment for Commitment Schemes. Let A be a (non-uniform) probabilistic Turing machine, specifying a man-in-the-middle strategy. A runs in time polynomial in the security parameter n. Let $z \in \{0,1\}^*$ be an arbitrary string (denoting the non-uniform "advice" for A). Further let $v \in \{0,1\}^n$ be an arbitrary string. The experiment begins by selecting uniform randomness for A, and honest parties C and R. $A(z)$ interacts with C, receiving a commitment to the string v, and acting as a receiver; A simultaneously participates in another interaction with an honest receiver R, acting as a committer. A's interaction with C is called the *left* interaction, and its interaction with R is called the *right* interaction. Let $\tilde{v}$ denote the string committed by

A in the right execution.[5] As soon as A completes its interaction on right, value $\tilde{v}$ is provided to A—irrespective of the state of left interaction.[6]

We prefer to work with *tag* based definitions; and hence every execution of $\langle C, R\rangle$ defines a tag string. We stick to the simple class of protocols where the tag string is chosen by the receiver.[7] Let the tag string on right be $\mathtt{tag}$, and $\tilde{\mathtt{tag}}$ on left (chosen by A). Without loss of generality, we assume that the length of the tag strings is the same as the security parameter n. If $\tilde{\mathtt{tag}} = \mathtt{tag}$, A's interaction on right is set to the abort symbol $\perp$.

The *joint* view (of A and R) in the experiment consists of: the randomness of A, the auxiliary input z, all messages A receives in *both* interactions, the value $\tilde{v}$, *and* the randomness of R. The experiment is a random process, and hence the joint view of A is a random variable. For $v \in \{0,1\}^n, z \in \{0,1\}^*, \mathtt{tag} \in \{0,1\}^n, n \in \mathbb{N}$, the joint view in the experiment (which includes $\tilde{v}$ given to A after completion of right side commitment) is denoted by:

$$\mathsf{mim}^{A}_{\langle C,R\rangle}(v, z, \mathtt{tag})$$

Definition 1 (Non-malleable Commitments) *A commitment scheme $\langle C, R\rangle$ is said to be non-malleable if for every non-uniform probabilistic polynomial time Turing machine A (man-in-the-middle), for every pair of strings $v_0, v_1 \in \{0,1\}^n$, every tag-string $\mathtt{tag} \in \{0,1\}^n$, every (advice) string $z \in \{0,1\}^*$, the following two random variables are computationally indistinguishable,*

$$\mathsf{mim}^{A}_{\langle C,R\rangle}(v_0, z, \mathtt{tag}) \quad \text{and} \quad \mathsf{mim}^{A}_{\langle C,R\rangle}(v_1, z, \mathtt{tag})$$

In general, in many schemes, tags of smaller length such as $\log n$ are also allowed. Ideally we want schemes with longer tag length such as n.

Man-in-the-Middle Experiment for Interactive Proofs. Let A be a (non-uniform) probabilistic Turing machine, specifying a man-in-the-middle strategy. A runs in time polynomial in the security parameter n. Let $z \in \{0,1\}^*$ be an arbitrary string (denoting the non-uniform "advice" for A). Let $\langle P, V\rangle$ be an interactive proof system for an $\mathcal{NP}$ complete language L. Let A and z be as before (in the experiment for non-malleable commitments). Let $x \in L$ be a statement of length n; we assume that P is polynomial time and receives a witness $w \in R_L(x)$ as its auxiliary input. The experiment begins by selecting uniform randomness for A, and honest parties P and V. $A(x, z)$ interacts with $P(x, w)$ on left acting as a verifier in the proof for $x \in L$; A simultaneously participates in a right proof with V, proving a related statement $\tilde{x}$, supposedly in L. At some point, all parties halt.[8] As before, let the tag strings on left and right be $\mathtt{tag}$ and $\tilde{\mathtt{tag}}$ respectively with $|\mathtt{tag}| = |\tilde{\mathtt{tag}}| = n$, and if $\mathtt{tag} = \tilde{\mathtt{tag}}$, right interaction is set to $\perp$.

The *joint* view (of A and V) in the man-in-the-middle experiment consists of: statements $x, \tilde{x}$, the randomness of A, the auxiliary input z, all messages A receives in *both* interactions, *and* the

[5] Value $\tilde{v}$ always exists, possibly set to $\perp$ if A aborts or if A is not committed to a single value.

[6] Note that this is a slight deviation from standard definition, which provide A the value $\tilde{v}$ only at the end of both interactions. Nevertheless, all non-malleable commitment schemes that we know of, are not affected by this change.

[7] Once again, this choice is in the interest of a cleaner exposition of our results, and without loss of generality. Note that A is free to choose the tag in the left session *adaptively*, after viewing the tag chosen by the receiver in the right session.

[8] String $\tilde{x}$ may be chosen either *adaptively* depending on the left execution, or *statically* by announcing it before the left execution begins. More precisely, we say that A chooses $\tilde{x}$ statically, if A on input (x, z) outputs $\tilde{x}$. Then the experiment proceeds as described.

randomness of V. The experiment is a random process, and hence the joint view of A is a random variable. For $x \in L$, $w \in R_L(x)$, $z \in \{0,1\}^*$, $\mathtt{tag} \in \{0,1\}^n$, $n \in \mathbb{N}$, the joint view in the experiment (sometimes called the *real* experiment) is denoted by:

$$\mathsf{view}^A_{\langle P,V\rangle}(x, w, z, \mathtt{tag})$$

When a random variable representing a view, e.g., $\mathsf{view}^A_{\langle P,V\rangle}(x, w, z, \mathtt{tag})$ is *accepting*, we abuse the notation and write: "$\mathsf{view}^A_{\langle P,V\rangle}(x, w, z, \mathtt{tag})$=1". We now recall two formulations of non-malleability from literature: non-malleable zero-knowledge and simulation-extractability.

Informally, a system $\langle P, V\rangle$ is non-malleable, if for every man-in-the-middle adversary A, there exists a stand-alone machine A^* such that A^* convinces an external verifier with essentially the same probability as A would in the real experiment. Let $\mathsf{mim}^A_{\langle P,V\rangle}(x, w, z, \mathtt{tag})$ and $\mathsf{sta}^{A^*}_{\langle P,V\rangle}(x, z, \mathtt{tag})$ be random variables that represent the output of V in the man-in-the-middle execution and the stand-alone execution respectively. Here $z \in \{0,1\}^*$ is the auxiliary input of A. Let $\langle A^*, V\rangle(x, z, \mathtt{tag})$ denote the view of V in a random execution of the proof system defined by $\langle P, V\rangle$, where A^* plays the role of the prover.

Definition 2 (Non-malleable Interactive Proofs) *An interactive proof system $\langle P, V\rangle$ for a language L is said to be* non-malleable *if for every* PPT *Turing machine A (man-in-the-middle), there exists a* PPT *Turing machine A^* (stand alone prover), such that for every $x \in L$, every $w \in R_L(x)$, every tag string $\mathtt{tag} \in \{0,1\}^n$, every polynomial $q(\cdot)$, every (advice) string $z \in \{0,1\}^*$, and every sufficiently large $n \in \mathbb{N}$,*

$$\Pr\left[\mathsf{view}^A_{\langle P,V\rangle}(x, w, z, \mathtt{tag}) = 1\right] < \Pr\left[\langle A^*, V\rangle(x, z, \mathtt{tag}) = 1\right] + \frac{1}{q(n)}$$

If, in addition, $\langle P, V\rangle$ is also zero knowledge*, then $\langle P, V\rangle$ is said to be a* non-malleable zero-knowledge *interactive proof system.*

Note that, a non-malleable interactive proof is not necessarily zero knowledge. A somewhat different formulation is that of simulation-extractability (which implies non-malleable zero-knowledge). This notion requires a simulator (as in $\mathcal{ZK}$), which simulates the joint view, and also outputs a witness for the right-hand side statement if V accepts on right.

Definition 3 (Simulation Extractable Interactive Proofs) *An interactive proof system $\langle P, V\rangle$ for a language L is said to be* simulation extractable *if for every* PPT *Turing machine A (man-in-the-middle), there exists a probabilistic expected polynomial time Turing machine SE, such that for every $x \in L$, every $w \in R_L(x)$, every tag string $\mathtt{tag} \in \{0,1\}^n$, and every (advice) $z \in \{0,1\}^*$, the following properties hold:*

1. *Joint-view in the real experiment, $\mathsf{view}^A_{\langle P,V\rangle}(x, w, z, \mathtt{tag})$, is computationally indistinguishable from the first output (i.e., simulated joint-view) of SE, denoted by $SE_1(x, z, \mathtt{tag})$.*
2. *If the right execution in $SE_1(x, z, \mathtt{tag})$ is accepting for an instance $\tilde{x} \in \{0,1\}^n$, then the second output of SE is a string $\tilde{w}$ such that $\tilde{w} \in R_L(\tilde{x})$.*

We note that we will use the words argument and proofs interchangeably, since our focus here is to construct interactive arguments only [BCC88], but the definitions are general and apply to both. We also note that simulation-extractability is motivated by much prior work, in particular see DeSantis et al. [DDO+01], and witness-extended emulation suggested by Lindell [Lin01].

3 Efficient Simulation-Sound Interactive Proofs

This section presents our first main result, by constructing a simulation-sound protocol based only on black-box use of non-malleable commitment scheme $\langle C, R\rangle$. The notion of simulation-soundness, in the interactive setting in *plain* model, despite implicit discussions in many works, has never appeared formally. In the common reference string (CRS) model, it was introduced by Sahai [Sah99] for non-interactive $\mathcal{ZK}$; and extension for interactive case—still in the CRS model—was considered by Garay, MacKenzie, and Yang [GMY06]. Building upon these works, we present a formulation in the plain model.

3.1 The Definition

Intuitively, simulation soundness means that in a $\mathcal{ZK}$proof, a man-in-the-middle adversary A cannot produce a convincing proof for a false statement, *even if* it can see *simulated* proofs for statements of its own choice, including *false* statements. Note that since A has to be able to access simulated proofs, in some sense, this automatically guarantees zero knowledge. This is unlike non-malleable interactive proofs, which may or may not be zero knowledge. The formal definition requires a single machine S—the simulator—which guarantees indistinguishability of the view for true statements, and the soundness for statements on *right hand side* even in the presence of simulated false proofs on *left hand side*.

Definition 4 (Simulation Sound Interactive Proofs) *An interactive proof system $\langle P, V\rangle$ for a language L is said to be* simulation sound *if for every probabilistic polynomial time Turing machine A (man-in-the-middle), there exists a probabilistic expected polynomial time machine S such that,*

1. *For every $x \in L$, every $w \in R_L(x)$, every $\mathtt{tag} \in \{0,1\}^n$, and every (advice) $z \in \{0,1\}^*$, the following distributions are computationally indistinguishable,*

$$S(x, z, \mathtt{tag}) \quad \textit{and} \quad \mathsf{view}^A_{\langle P,V\rangle}(x, w, z, \mathtt{tag})$$

2. *For every $x \in \{0,1\}^n$, every $\mathtt{tag} \in \{0,1\}^n$, every (advice) $z \in \{0,1\}^*$, every polynomial $q(\cdot)$ and every sufficiently large $n \in \mathbb{N}$,*

$$\Pr\left[\nu \leftarrow S(x, z, \mathtt{tag}); \tilde{x} \notin L \wedge \tilde{b} = 1\right] < \frac{1}{q(n)}$$

where, $\tilde{x}$ represents the right hand side statement and $\tilde{b}$ denotes whether the right hand side verifier accepts, in the simulated joint view ν.

Remarks.

1. The second requirement in our definition is analogous to the requirement in [Sah99, GMY06]. This is essentially the core requirement of the definition. It also makes the definition somewhat *strict* in the sense that A can receive simulated proofs for statements that it *knows* are false. However, this is unavoidable: while A might know that a particular statement is false, it is not clear how the simulator S can access this knowledge. As a result, S must be able to simulate for all false x, irrespective of what A knows about x. This is the fact we exploit in constructing a counter-example and prove that SimExt $\not\Rightarrow$ SimSound. The first requirement of the definition is for capturing $\mathcal{ZK}$.

2. The strict requirement can be relaxed by considering a *distribution* based definition in which a sampler samples a statement (either true or false) but A does not know which is the case. A then receives proofs only for such sampled statements. This yields a weaker definition, denoted SIMSOUND_D. In this case, it is straightforward to see that $\text{SIMEXT} \Rightarrow \text{SIMSOUND}_D$. We remark that this is indeed usually the situation in cryptographic settings.

On the counter-example. In the light of our counter-example (see Section 4.2), it might be tempting to think that this definition is not so natural, since a "good" definition should be implied by simulation-extractability. The counter-example may then look like a mere artifact of our definition and leading to the belief that SIMSOUND_D is the "right" definition. We strongly disagree with this thought, and argue that this is not the right way to think about simulation-soundness. We believe that SIMSOUND should be seen as an independent notion and the "right" definition based on one's intuition should be formulated. In this sense, our formulation is just a natural extension of [Sah99, GMY06] to the plain model.

Indeed, SIMSOUND has no "obligation" to be consistent with SIMEXT; if it so happens that an independent and intuitive formulation, aimed at conceptual understanding of SIMSOUND, turns out to be consistent (or not, for that matter) with SIMEXT, then so be it. An interesting parallel can be drawn with *strong witness-indistinguishability* (WI) [Gol01]. Strong WI is sufficient for "natural" cases that arise in practice where the definition, very informally, only considers *indistinguishable distributions* over the statements. Whereas, when considering *zero knowledge*, the right definition must still deal with *individual statements*—much like the current formulation of SIMSOUND.

3.2 Tool: Extractable Commitment

Consider the following simple challenge-response based bit commitment scheme $\langle \widehat{C}, \widehat{R} \rangle$, which can be based on top of any standard bit commitment scheme $\langle C', R' \rangle$ (e.g., [Nao89]). The scheme has been used in several works, starting from [PRS02, Ros04]. The protocol $\langle \widehat{C}, \widehat{R} \rangle$ for committing to a bit b is described in Figure 1.

Commit Phase:

1. To commit to a bit b, $\widehat{C}$ chooses n independent random pairs $\{b_i^0, b_i^1\}_{i=1}^n$ of bits such that $b_i^0 \oplus b_i^1 = b$; and commits to them to $\widehat{R}$ using $\langle C', R' \rangle$. Let $c_i^0 \leftarrow C'(b_i^0)$ and $c_i^1 \leftarrow C'(b_i^1)$ for every $i \in [n]$.
2. $\widehat{R}$ sends n uniformly random bits $r_1, \ldots, r_n$.
3. For every $i \in [n]$, if $r_i = 0$, $\widehat{C}$ opens c_i^0, otherwise it opens c_i^1 to $\widehat{R}$ by sending the decommitment information (according to $\langle C', R' \rangle$).

Open Phase: $\widehat{C}$ opens all the unopened commitments by sending the decommitment information for each of them.

Figure 1: Extractable Commitment Scheme $\langle \widehat{C}, \widehat{R} \rangle$

Some remarks about $\langle \widehat{C}, \widehat{R} \rangle$ are in order.

1. If $\langle C', R' \rangle$ is statistically binding, so is $\langle \widehat{C}, \widehat{R} \rangle$. We adopt the convention that if not all shares are such that $b_i^0 \oplus b_i^1 = b$, then the adversary has essentially aborted, setting $b = \perp$. It suffices for our application.

2. To commit to a string $s = s_1, s_2, \ldots, s_n$, execute n instances of $\langle \widehat{C}, \widehat{R} \rangle$, in *parallel*—one for each bit s_i. This is a string commitment scheme, which we shall denote by $\langle \widehat{C}_s, \widehat{R}_s \rangle$. Note that $s = \perp$ if $s_i = \perp$ for any $i \in [n]$.

3. Scheme $\langle \widehat{C}_s, \widehat{R}_s \rangle$ is a public coin protocol for $\widehat{R}_s$.

4. Finally, $\langle \widehat{C}_s, \widehat{R}_s \rangle$ admits an extractor algorithm $CExt$ which extracts a string $\texttt{str}'$ from every C^* such that if C^* commits to a valid string $\texttt{str}$, then $\texttt{str}' = \texttt{str}$. The running time of $CExt$ is inversely proportion to the probability that C^* makes a convincing commitment to R. If we increase the challenge-response rounds to $\omega(1)$, then $CExt$ runs in polynomial time. A formal claim about the extractor is given in appendix B.1.

3.3 The Simulation-Sound Protocol

We are now ready to present our simulation-sound interactive *argument* system $\langle P, V \rangle$. Our protocol uses a *non-malleable commitment scheme* $\langle C, R \rangle$, and also the *bit* commitment scheme $\langle \widehat{C}, \widehat{R} \rangle$ described above. These schemes are used in a black-box manner.

At a high level, our protocol is essentially a parallel repetition of Blum's protocol for Graph Hamiltonicity [Blu87] with the following modifications: (a) instead of using a standard perfectly binding commitment scheme, the prover P uses the 3-round commitment scheme $\langle \widehat{C}, \widehat{R} \rangle$ described above. (b) Further, similar to [GK96], we require the verifier to commit to its challenge in advance. However, unlike [GK96] where a statistically hiding commitment scheme is used, the verifier V in our protocol uses the non-malleable commitment scheme $\langle C, R \rangle$ to commit to its challenge.

Protocol $\langle P, V \rangle$. We now formally describe the protocol. Let P and V denote the prover and verifier respectively. The common input is a graph G in the form of a $n \times n$ adjacency matrix. Further, the private input to P is a Hamiltonian cycle H in G. The protocol proceeds as follows:

1. V chooses a uniformly random n bit string ch, and commits to it by using the non-malleable commitment scheme $\langle C, R \rangle$, where V acts as C and P acts as R.

2. For every $i \in [n]$:

   (a) P chooses a random permutation π_i (a string of size $2n$) and prepares $G_i = \pi_i(G)$.

   (b) P commits to each bit $b_{i,j}$ in matrix G_i and each bit $b'_{i,k}$ in π_i to V using the commitment scheme $\langle \widehat{C}, \widehat{R} \rangle$,[9] for every $j \in [n^2]$ and $k \in [2n]$.

   Note that this step can be seen as running $2n$ parallel instances of $\langle \widehat{C}_s, \widehat{R}_s \rangle$, n of which commit to n^2 size bit strings $G_1, \ldots, G_n$ and and the remaining n commit to $2n$ size bit strings $\pi_1, \ldots, \pi_n$.[10] Alternatively, this step is essentially equivalent to committing a string $\texttt{str}$ of size $n^3 + 2n^2$ using the protocol $\langle \widehat{C}_s, \widehat{R}_s \rangle$.

3. V sends ch, and decommits by sending the corresponding decommitment information.

[9] Each matrix G_i has n^2 cells, and each cell contains a single bit.

[10] Every permutation π_i for G with n vertices can be encoded using a string of length at most $2n$ simply by listing the new "name" of every vertex under the permutation π_i, where "name" is just an index in $[n]$ with $\log n$ size representation

4. For every $i \in [n]$:
   (a) If $ch_i = 0$, P reveals π_i and G_i, and decommits according to $\langle \widehat{C}, \widehat{R} \rangle$.
   (b) Otherwise, P reveals the Hamiltonian Cycle in G_i by sending the corresponding decommitment information.

Facts about $\langle P, V \rangle$.

1. If graph G does *not* contain a Hamiltonian cycle, and the commitments in step 2 define a unique string (say `str`), then after this step, there is *at most one* challenge string ch for which a cheating prover P^* can provide a convincing answer in step 4.[11]

2. Furthermore, in the above case, if there does exist a challenge ch for which a convincing answer can be produced after step 2, then given string `str`, one can reconstruct the challenge ch in the following manner: for every $i \in [n]$, let π'_i denote the i^{th} permutation and G'_i denote the i^{th} graph in string `str`. Then, if $\pi'_i(G) = G'_i$, set $ch_i = 0$, else set $ch_i = 1$.

3.4 Proof of Security

Theorem 1 *Protocol $\langle P, V \rangle$ is a simulation-sound interactive argument system for Graph Hamiltonicity, as per definition 4.*

To prove Theorem 1, we need to demonstrate an expected polynomial time machine S (simulator), satisfying both properties in definition 4. Our simulator S is almost identical to the Goldreich-Kahan simulator [GK96], with some obvious modifications. Before we describe our simulator, we first define a machine A' that is used later in our description.

Machine A'. We define A' as a machine that internally incorporates the adversary A and honest verifier V. A' will act as a cheating verifier and receive a proof from an honest prover. Internally, A' forwards this proof to A as the left interaction of $\langle P, V \rangle$. In addition,A' also simulates the right interaction for A, by letting it interact with the internal verifier V. At the end of the execution, A' separates the left and the right views to represent the joint-view (of the man-in-the-middle experiment for interactive proofs). A' produce such a joint-view simply by replaying the messages with fixed randomness that it used for V and A internally. The reason, we create this artificial machine A' is because we can now run Goldreich-Kahan simulator on it.

Simulator S. The description of our simulator S, for proving SimSound, appears in Figure 2. Every time S rewinds A' to a previous point in execution, it provides A' with fresh randomness for the rest of the execution.

It is instructive to note that while in the NmCom-experiment, the adversary can obtain $\tilde{v}$ as soon as it finishes the right session, no such "luxury" is available in NmZK-experiment. Therefore,

[11] Specifically, consider the simpler case of the 3-round Blum Hamiltonicity (BH) protocol (the argument for our scheme follows in a similar manner). Recall that in BH, if the challenge bit is 0, then the prover P must open the commitments (sent in the first step) to a permutation π and a graph G' such that $\pi(G) = G'$. Otherwise, if the challenge bit is 1, P must reveal a Hamiltonian cycle in the committed graph (without decommitting to the entire committed graph). Then, if graph G does not contain a Hamiltonian cycle, and a cheating prover manages to succeed in the last step irrespective of whether the challenge bit is 0 or 1, we can contradict the binding property of the commitment scheme.

Let A' denote be a machine that internally incorporates adversary A and honest verifier V as described earlier. The simulator S first fixes a uniformly random tape for A' and then proceeds according to the following steps:

S1. S simulates the first step of the protocol by playing an honest receiver R in the execution of the non-malleable commitment scheme $\langle C, R\rangle$ with the adversary A'. Since $\langle C, R\rangle$ is statistically binding, except with negligible probability, A' is committed to a unique challenge string. Let the state of A' at the end of this step be $st_{A'}$, which S records.

S2. S now plays the next round of the protocol but commits to dummy strings (e.g., all 1s). That is, S commits by executing $2n$ parallel instances of $\langle \widehat{C}_s, \widehat{R}_s\rangle$: n instances of n^2-size strings (representing graphs) and the rest of size $2n$ (representing permutations). At the end of this step, if A' successfully opens a challenge string ch for step S1 commitment, S records ch and proceeds to the next step. Otherwise, S outputs the current view of A' and halts.

S3. S repeats step (S2) with fresh randomness until it records n^2 valid openings to ch. If t is the total number of trials, then let $\bar{p} = n^2/t$ be an estimate of $p(G, r)$. Here $p(G, r)$ is the probability that A' (when initialized with state $st_{A'}$) successfully opens ch in step (S2) over the randomness of step (S2).

S4. S now reinitializes A' with the same state $st_{A'}$, and plays step (S4) as follows. Recall that ch is the string opened in step (S2) by A. For every $i \in [n]$, S does the following: if $ch_i = 0$, S chooses a random permutation π_i and commits to π_i and $G_i = \pi_i(G)$; otherwise, S commits to a random permutation and a random n-cycle graph H_i (i.e., an $n \times n$ adjacency matrix where the cells corresponding to the cycle are set to 1 while the other cells are set to random bits). If A' replies by correctly revealing the string ch, S proceeds to complete the simulation by correctly revealing the openings to the commitments. Otherwise, the entire step (S4) is repeated for at most $\frac{\text{poly}(n)}{\bar{p}}$ times, until A' correctly reveals ch. If all the attempts fail, S outputs a special symbol indicating time-out.

Figure 2: Simulator S for $\langle P, V\rangle$.

A—who works in the NmZK-experiment—never asks for any such values. Note that, step (S1) represents a partial "main thread" of execution of S, which it completes in step (S4) if A' opens the challenge successfully.

Proving simulation-soundness. To prove that $\langle P, V\rangle$ satisfies definition 4, we show that it satisfies its two requirements. The first requirement is akin to proving that $\langle P, V\rangle$ is a $\mathcal{ZK}$ interactive argument system. The proof for this is identical to [GK96], barring some trivial changes, and is therefore omitted. In particular, it follows almost immediately if we show that time-out is output with negligible probability (see [GK96] or the proof of Claim 2 in section 4.1 where a similar claim is argued).

We now focus on the second requirement in definition 4, which requires that A cannot prove a false statement in the view output by S, *even if* the input to S is a false statement: i.e., an arbitrary graph $G \in \{0,1\}^{n^2}$, which may not be Hamiltonian.[12]

Assume that the second requirement does not hold. Then, there exists a (non-uniform) PPT man-in-the-middle adversary A, a polynomial $q(\cdot)$, an advice string $z \in \{0,1\}^*$, and infinitely many $n \in \mathbb{N}$ such that for every n there exists a graph $G \in \{0,1\}^{n^2}$, and a tag string $\mathtt{tag} \in \{0,1\}^n$ such

[12]For simplicity, we are dealing with a n^2 size statement here, but this is only a syntactic change, and can be removed by scaling the number of vertices in G down to $\sqrt{n}$.

that over the randomness of S:

$$\delta(n) \stackrel{\text{def}}{=} \Pr\left[\nu \leftarrow S(G, z, \mathtt{tag}); \tilde{G} \notin L \wedge \tilde{b} = 1\right] > \frac{1}{q(n)}$$

where, $\tilde{G}$ represents the right hand side instance and $\tilde{b}$ denotes whether the right hand side verifier accepts, in the simulated joint-view ν. Fix one such n, along with statement $G_n \in \{0,1\}^{n^2}$, and tag string $\mathtt{tag}_n \in \{0,1\}^n$, and let

$$\delta_n = \Pr\left[\nu \leftarrow S(G_n, z, \mathtt{tag}_n); \tilde{G} \notin L \wedge \tilde{b} = 1\right]$$

which is larger than $1/q(n)$. Now, recall that the man-in-the-middle A controls the scheduling of messages in the left and right executions. Figure 3 describes three representative schedules that A can choose from. Since the overall success probability of A is δ_n, A must succeed in one of the schedules with probability at least $\frac{\delta_n}{3}$. For each schedule, we will show how to break some security property of scheme $\langle C, R\rangle$.

We start by describing the three schedules, as shown in Figure 3.

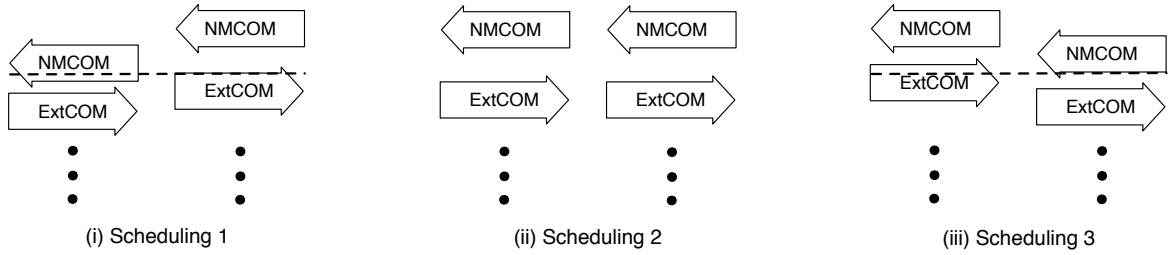

Figure 3: Three schedules for A. NMCOM $:= \langle C, R\rangle$ and EXTCOM $:= \langle \hat{C}_s, \hat{R}_s\rangle$

Scheduling 1. A starts the right execution of extractable-commitment part EXTCOM $:= \langle \hat{C}_s, \hat{R}_s\rangle$ before its left execution. In particular, this means that A sends out the first message of $\langle \hat{C}_s, \hat{R}_s\rangle$ in the right execution without seeing the corresponding message in the left execution.

Scheduling 2. A schedules the left and right executions in a synchronous manner.[13]

Scheduling 3. A forces the EXTCOM protocol $\langle \hat{C}_s, \hat{R}_s\rangle$ in the left execution to be started before the NMCOM protocol $\langle C, R\rangle$ is completed in the right execution.

In scheduling 1, we will break the hiding property of $\langle C, R\rangle$, while in schedules 2 and 3, we will break the non-malleability property of $\langle C, R\rangle$. We now give more details.

Proof for scheduling 1. The main observation is that in this scheduling A has already fixed the message for which he can succeed, without any help from the simulator. Therefore, we can extract this value from the EXTCOM-phase. Very briefly, the fact that simulator rewinds on left is not a problem because the extractor for EXTCOM is public-coin and S never rewinds to a point higher than EXTCOM on left. Details follow.

[13]This means that for every $i \in [r]$, where r is the number of rounds in $\langle P, V\rangle$, the i^{th} message of the protocol in left and right executions is sent only after the $(i-1)^{th}$ message of the protocol has been sent in both left and right executions.

We will construct a cheating receiver R^* such that if A succeeds with probability p, then R^* breaks the hiding property of $\langle C, R\rangle$ with probability at least poly(p). We first consider the following (simulator) machine S' that will later be used in the description of R^*.

SIMULATOR S'. Let v_0, v_1 be two distinct n-bit strings, and let $z' = (v_0, v_1, z)$. Simulator S' internally incorporates the man-in-the-middle A (which expects executions of $\langle P, V\rangle$ on both sides), graph G_n, string z', and tag string $\mathtt{tag}_n$. It now proceeds exactly as our simulator S, except for the following differences:

1. Instead of internally emulating the actions of V internally up to the second phase (i.e., the EXTCOM-phase), machine S' expects them to be coming from an external party.

2. S' halts as soon as the second-stage completes in the right session (i.e., A completes its EXTCOM-stage on the right).

We are now ready to describe the cheating receiver R^*. Let $p = 1/\text{poly}(n)$ be success probability of A in scheduling 1.

RECEIVER R^*. The cheating receiver R^* interacts with an honest committer C who commits to v_b (for a randomly chosen bit b), and outputs b as its guess of b. Machine R^* internally incorporates the program of machine S' as described above. It starts an internal execution of S', and when S' asks for the commitments corresponding to the non-malleable commitment phase on right, R^* interacts with an external committer for the NMCOM-scheme $\langle C, R\rangle$ and replays its messages to A'. As soon as the last message by this external committer has been sent, R' disconnects with the external committer, and continues the execution up to the point where S' sends the first message of EXTCOM-scheme. At this point, it does the following:

1. R^* records the state of S' at this point. Call this machine C^*. Note that machine C^* represents a (cheating) committer algorithm for the EXTCOM-scheme, who has already sent its first message.

2. R^* chooses a random strings r_0 and feeds it to C^*, and then continues the execution of C^* from here onwards. If C^* needs randomness, R^* provides it to him. If C^* expects more strings of the form r_0 (since internal rewindings on left may go past this challenge), R^* feeds C^* with fresh challenges. At some point, C^* produces a response. If no response is received even after $4T/p$ steps, R^* outputs a random bit and halts. Here T is the expected running time of S'.

3. R^* repeats the above step with a fresh challenge r_1.

4. Note that if (valid) answers for both challenges are received, R^* can compute the value of v_b (simple $\oplus$, see section 3.2).

Observe that due to scheduling 1, and construction of S, up to this point, no rewindings has been performed on left (internally). And therefore external committer is never rewound by R^*. From here on, the analysis for success probability of R^* is quite standard. First observe that execution of R^* up to the point where it constructs C^* is identical to that of S' (and hence S). Therefore, it holds—by a standard averaging argument—with at least $p/2$ probability, C^* commits to a value which it will successfully open with probability at least $p/2$ over the rest of the randomness of

the execution. In addition, since the overall expected running time is T, for any $p/2$-fraction large set of transcripts, not all transcripts in this set can have running time more than $2T/p$. In particular, for each such set, no more than half of the transcripts in the set can have running time $2 \cdot 2T/p$. It follows that C^* responds in time less than $4T/p$ with probability at least half conditioned on the aforementioned prefix of executions. Therefore, valid responses for both r_0 and r_1 are obtained within time $4T/p$ w.p. $(p/2 \times 1/2)^2 = p^2/16$. Since such a suitable prefix occurs with probability $p/2$, R^* predicts b with following probability or more: $q + (1-q)/2 = 1/2 + q/2$ where $q = p/2 \cdot p^2/16 = p^3/32$.

Proof for schedule 2. In this case, we will demonstrate a PPT man-in-the-middle adversary A' for $\langle C, R\rangle$ and an expected PPT distinguisher D' violating the non-malleability property of $\langle C, R\rangle$. The key-idea for this schedule is that the value that A commits in NmCom, will be given as input along with the joint-view. And therefore, we do *not rewind* A at all, since we already know the trapdoor for simulation. We proceed by first constructing in A' for the commitment, and then a distinguisher, who simply "restarts" from the point where A' stopped and continues the simulation until the ExtCOM on right concludes. Details follow.

Man-in-the-middle A'. Let v_0, v_1 be two distinct n-bit strings, and let $z' = (v_0, v_1, z)$. Adversary A' internally incorporates machine A (which expects executions of $\langle P, V\rangle$ on both sides), graph G_n, string z', and tag string $\mathtt{tag}_n$. It now proceeds exactly as our simulator S, except for the following differences:

1. In step (S1), instead of internally emulating the actions of committer C (on behalf of right-hand-side V), A' receives this commitment from an external committer who is committing to v_b, for a randomly chosen bit b (outside the view of A'). Likewise, instead of internally emulating R (on behalf of left-hand-side P), it forwards the interaction to an external receiver R.[14]

2. A' halts as soon as A finishes its first step commitment on left.

Note that the state of A' after this execution, denoted $\mathrm{st}_{A'}(G_n, z', \mathtt{tag}_n)$, along with the view of R, contains a joint-view $\mathsf{mim}^A_{\langle C,R\rangle}(v_b, z', \mathtt{tag}_n)$. Let $\tilde{v}$ be the value committed to on right by A in this joint view. We now describe the distinguisher D', who gets $(\tilde{v}, \mathsf{mim}^A_{\langle C,R\rangle}(v_b, z', \mathtt{tag}_n))$ as input, and predicts bit b with probability $1/2 + \delta_n/4$.

Distinguisher D'. The distinguisher D' internally incorporates A' (or A, since A' and A are essentially identical; see above), graph G_n, string z', and tag-string $\mathtt{tag}_n$. D' gets as input a tuple $(\tilde{v}, \mathsf{mim}^A_{\langle C,R\rangle}(v_b, z', \mathtt{tag}_n))$.[15]

D' first prepares a committer C^* for the commitment scheme $\langle \widehat{C}_s, \widehat{R}_s\rangle$ as follows. The machine C^* is essentially identical to our simulator S, except for the following differences:

1. C^* does not execute step (S1). Instead, it initializes A with the adversary's view in the input joint view $\mathsf{mim}^A_{\langle C,R\rangle}(v_b, z', \mathtt{tag}_n))$. Specifically, C^* first fixes A with the same (adversary) random tape as in the view $\mathsf{mim}^A_{\langle C,R\rangle}(v_b, z', \mathtt{tag}_n))$. Then, C^* (without using any

[14] When using black-box version of Goyal's scheme, only the part corresponding to a "small" tag will be received from the outside committer. Details are given in appendix A.

[15] Note that if $\langle C, R\rangle$ only satisfies the weaker notion of *non-malleability w.r.t. replacement* [Goy11], then the distinguisher D' gets a tuple $(\tilde{rv}, \mathsf{mim}^A_{\langle C,R\rangle}(v_b, z', \mathtt{tag}_n))$ from a distribution "compatible" with $(\tilde{v}, \mathsf{mim}^A_{\langle C,R\rangle}(v_b, z', \mathtt{tag}_n))$. See appendix A for more details.

fresh randomness) replays all the honest committer and receiver messages in the joint-view $\mathsf{mim}^A_{\langle C,R\rangle}(v_b, z', \mathtt{tag}_n)$) to A. The view corresponding to honest receiver is the view S would have internally simulated in (S1) on behalf of the prover, and the view corresponding to the honest committer is the view S would have internally simulated on right, in step (S1). A's view is identically distributed as per the adversary's view in $\mathsf{mim}^A_{\langle C,R\rangle}(v_b, z', \mathtt{tag}_n)$). At the end of this initialization, C^* is exactly like the simulator at the end of step (S1).

2. C^* does not execute steps (S2) and (S3) (since it already has string $\tilde{v}$ as input). Note that unlike S, C^* does *not* rewind A.

3. Further, C^* executes step (S4) with the challenge string $ch = \tilde{v}$ (with current state of A)[16]. If A responds by correctly revealing the string $\tilde{v}$, C^* proceeds in the same manner as S; otherwise it stops the execution. However, unlike S, C^* executes this step (S4) only once, and never repeats it.

4. Finally, consider the execution of the commitment protocol $\langle \widehat{C}_s, \widehat{R}_s\rangle$ in Stage 2 in the right session. Instead of internally simulating an honest receiver (like S does), C^* expects this to be an external party $\widehat{R}_s$. Thus, all the messages from A for this stage are forwarded externally to $\widehat{R}_s$, and vice-versa. C^* halts as soon as this phase ends. The view is replaced by $\perp$ if $\widehat{R}_s$ does not accept the last committer message from C^*.

If C^* does not output $\perp$, then D' applies the extractor $CExt$ for the commitment scheme $\langle \widehat{C}_s, \widehat{R}_s\rangle$ on machine C^* with the view obtained by $\widehat{R}_s$, and extracts a string $\mathtt{str}$.[17] Then, from string $\mathtt{str}$, D' constructs a challenge string $\tilde{ch}$ in the same manner as described earlier in the fact 2 for $\langle P, V\rangle$.[18] If $\tilde{ch} = v_0$, D' outputs 0. On the other hand, if $\tilde{ch} = v_1$, D' outputs 1. In any other case, D' outputs a random bit as its guess. This completes the description of the distinguisher D'.

Analysis of D'. We first analyze the running time of D'. Note that every step in D' is strict polynomial-time, except the step where the extractor $CExt$ is run. Now, let us consider the execution of the commitment protocol $\langle \widehat{C}_s, \widehat{R}_s\rangle$ in Stage 2 in the right session. Fixing the first committer message of the commitment protocol, let p be the probability that C^* (that internally incorporates A) produces a convincing last message for the receiver $\widehat{R}_s$. Here, the probability is taken over the random coins of the receiver and the random tape used by C^* (for interacting internally with A). Then, it follows from Claim 3 that the running time of $CExt$ is bounded by $\frac{\text{poly}(n)}{p}$. Now since D' runs the extractor $CExt$ with probability p, the total expected number of steps taken by D' are at most $\text{poly}(n) + p \cdot \frac{\text{poly}(n)}{p} = \text{poly}(n)$.

We now analyze the probability that D' correctly outputs bit b. Note that if the view sampled by C^* is one for which A is going to succeed in the right execution, then (at least) each bit in the extracted string $\mathtt{str}$ that A is required to open in the last step is well-defined. Specifically, let $\mathtt{str} = \mathtt{str}_1, \ldots, \mathtt{str}_n$. Then, if the i^{th} bit of string v_b is 0, then $\mathtt{str}_i$ must be equal to $\tilde{G}_i, \tilde{\pi}_i$ such that $\pi_i(\tilde{G}_i) = \tilde{G}$, where $\tilde{G}$ is the instance graph in the right execution. On the other hand, if the

[16] Hence it does not need state st_A

[17] Recall that C^* does not rewind A at any point in the left session and performs straight-line simulation. Then, if the left session gets rewound because of any possible rewindings performed by $CExt$, C^* simply re-uses the value $\tilde{v}$ during any possible re-execution of the left session.

[18] Recall that in the simulation-soundness experiment, we are only interested in the case where A manages to successfully prove a *false* theorem in the right execution. Therefore, the facts for $\langle P, V\rangle$ stated earlier are applicable here.

i^{th} bit of v_b is 1, then $\mathtt{str}_i$ must contain a permutation and an n-cycle (where the bits that do not correspond to the cycle may not be well-defined). Therefore, if A is going to succeed in the right execution, then the reconstructed challenge $\tilde{ch}$ must be equal to either v_0 or v_1.

Now recall that A is successful (in violating simulation soundness) with probability δ_n. This means, that if $CExt$ successfully extracts $\mathtt{str}$, then ch is indeed equal to v_b, and hence D' outputs a correct guess for b. Since extraction succeeds with $1 - \text{negl}(n)$ probability, we have that whenever A would have succeeded, the guess is correct with probability $\geq \delta_n - \text{negl}(n)$. For the case when the commitment from C^* is corresponding to a view where A would not have succeeded, the guess is still correct with probability $\frac{1}{2}(1 - \delta_n + \text{negl}(n))$. These two events are mutually exclusive, and hence by adding them we get that D' outputs a correct guess for b with probability at least $\frac{1}{2}(1 + \delta_n - \text{negl}(n)) \geq 1/2 + 1/4q(n)$. Hence the contradiction.

Proof for schedule 3. The proof for this schedule is identical to that for schedule 3. We first observe the following: in scheduling, A ends up completing his NmCom-phase before it has received the NmCom from V. The key-observation now is that since A's commitment finishes first, it can be given the value it has committed so far during the execution. Therefore, we consider a man-in-the-middle A' for NmCom, just as before; as soon as it receives the committed value (which is the trapdoor for simulation), it internally incorporates the distinguisher D' described above and runs it with this value. The proof therefore is essentially the same as for the case of schedule 2.

4 Relationship Between Various Notions of Non-Malleable ZK

In this section, we study the definitional equivalence between the three notions of non-malleability in the context of zero knowledge, namely, NmZK, SimSound, and SimExt. We obtain two results. First, in Section 4.1, we show that NmZK with the argument of knowledge (AOK) property implies SimExt. Then, in Section 4.2, we demonstrate that SimExt does not imply SimSound. Putting these results together, we obtain the following picture regarding the equivalence of the three notions:

$$\text{NmZK-AoK} \Leftrightarrow \text{SimExt} \not\Rightarrow \text{SimSound}$$

4.1 NmZK versus SimExt

Simulation-extractability was formulated in [PR05b], and it was shown that it implies non-malleable zero-knowledge. Perhaps somewhat surprisingly, we show that if the non-malleable zero-knowledge protocol is also an argument-of-knowledge, then it is actually simulation extractable.

Theorem 2 *If $\langle P, V\rangle$ is a* non-malleable zero knowledge argument of knowledge *for a language $L \in \mathcal{NP}$, it is in fact a* simulation extractable interactive argument *for L.*

This result is very generic, and applies to all types of protocols, be they computational or statistical $\mathcal{ZK}$, black-box or non-black-box $\mathcal{ZK}$, etc. This essentially shows that as far as the "moral" sense of non-malleability is concerned, the two definitions are equivalent. We note, however, that the theorem holds only for the *static* case where A must announce the theorem $\tilde{x}$ on input the theorem x, before the protocol execution begins. That is, $\tilde{x}$ depends only on x and not on the execution of the protocol. We do not consider the adaptive case (where A chooses $\tilde{x}$ based on the execution on left so far) in this paper, and leave it for future work.

Proof of Theorem 2. Let $\langle P, V \rangle$ be a non-malleable zero knowledge argument of knowledge for a language L with respect to tags of length n. This means that we have the following: (a) Since $\langle P, V \rangle$ is non-malleable, we have that $\forall$ PPT man-in-the-middle adversaries A, $\exists$ a stand-alone PPT adversary A^* such that it satisfies definition 2. Now, we can define probabilities p_a and p_{a^*} as follows: p_a is the probability that A convinces the verifier in the right session in the real execution, and p_{a^*} is the probability that A^* convinces an honest verifier. Note that by definition 2, we have that $p_{a^*} \geq p_a - \text{negl}(n)$. (b) Since $\langle P, V \rangle$ is zero knowledge, for every cheating verifier strategy, $\exists$ a simulator S satisfying the zero knowledge property. (c) Since $\langle P, V \rangle$ is an argument of knowledge, for every cheating prover P^*, $\exists$ an extractor E satisfying the argument of knowledge property.

Now, in order to prove Theorem 2, we need to construct a simulator-extractor SE satisfying Definition 3. Let V^* be a machine that internally incorporates the adversary A and verifier V and their interaction in the real execution.[19] Now, before proceed to formally describe our simulator-extractor, let us discuss a natural approach in order to highlight the main technical challenge.

Main Issue. Consider the following natural algorithm for a simulator-extractor SE: (a) Let V^* be a machine that internally incorporates the adversary A and verifier V and their interaction in the real execution. Then, first run the simulator S with V^* to output the joint view of A. (b) Now, since we are in the static case, the instance $\tilde{x}$ in the right execution does not depend on the left view. Then, if the joint view output by S contains an accepting right execution, run the extractor E on the stand-alone adversary A^* to extract a witness for $\tilde{x}$.

Now note that the running time of the second step in the above procedure is proportional to $\frac{p_s}{p_{a^*}}$, where p_s is the probability that the joint view output by S contains an accepting right execution. Unfortunately, $\frac{p_s}{p_{a^*}}$ is not necessarily bounded by a polynomial, hence the running time of the above procedure is not expected PPT. We note that this is reminiscent of a technical problem overcome by Goldreich and Kahan in the construction of constant round zero knowledge [GK96]. Indeed, we overcome this problem by using ideas from [GK96].

The simulator-extractor. We now give a detailed description of our simulator-extractor SE. The simulator-extractor SE receives statement x, advice string z, and tag-string $\mathtt{tag}$ as input. SE also receives access to either the code of A or black-box access to A depending upon the properties of A^* and S. SE runs the following steps:

S1. Run S with V^* on input x, z, and $\mathtt{tag}$ to generate a view view_{V^*}.[20] If the right execution in view_{V^*} is not accepting (i.e., A does not convince V), then output $(\mathsf{view}_{V^*}, \perp)$ and stop.

S2. Otherwise, let p_s denote the probability that simulator S in step (S1) outputs a view that contains an accepting right execution. Run S with V^* (on input x, z, and $\mathtt{tag}$) repeatedly

[19] Note that given the view of V^* in the real execution, we can construct the joint view of A (as defined in Section B) from the random tape of V^* and the messages that A and V exchange internally, as well as those exchanged between P and V^*. Likewise, if S is used with V^*, the joint view can still be constructed *even if* S is a non black-box simulator. This is done by first producing view_{V^*}, and then *replaying* the entire executions on left and right with the same V that was used to construct V^*. This yields messages corresponding to both left and right sessions of A, defining the complete joint view.

[20] SE simply runs the man-in-the-middle adversary A such that it interacts with S and V in the "left" and "right" executions respectively. In case S is a non black-box machine, SE is non black-box as well, since it will need the code of A to construct V^*.

until n^2 views with accepting right execution are obtained. Let $\overline{p_s} = \frac{n^2}{t}$ be the estimate of p_s, where t is the total number of trials.

S3. Run the following two procedures in *parallel*[21] and halt as soon as one of them finishes.

(i) Run the brute force (exponential-time) search procedure to compute a witness $\tilde{w}$ for $\tilde{x}$. Output the tuple $(\mathsf{view}_{V^*}, \tilde{w})$ and stop.

(ii) Run the following two steps: (a) First, run the (stand-alone) adversary A^* with an honest verifier V on inputs x, $\tilde{x}$ (where $\tilde{x}$ is the theorem proven by A in step S1), z, and tag repeatedly at most $\frac{\mathrm{poly}(n)}{\overline{p_s}}$ times until an accepting view view_{A^*} is generated. If no accepting view is generated after all the trials, then output a special symbol indicating $\mathsf{time\text{-}out}$ and stop. (b) Next, run the argument of knowledge extractor E on A^* with input view_{A^*} to extract a witness $\tilde{w}$ for $\tilde{x}$ from A^*. Output the tuple $(\mathsf{view}_{V^*}, \tilde{w})$ and stop.

This completes the description of our simulator extractor SE. We will now prove that it satisfies Definition 3. Here, the difficult part is to prove that the running time of SE is polynomial in expectation, as claimed below.

Claim 1 *The simulator-extractor SE runs in expected polynomial time.*

We further need to show that SE outputs a joint view indistinguishable from that in the real execution along with a valid witness for the instance in the right session. Towards that end, note that the correctness of the witness string output by SE follows directly from the correctness of the extractor machine E. Then, we only need to show that the joint view output by SE is indistinguishable from that in the real execution. To this end, we note that the proof for the same essentially follows from the zero-knowledge property guaranteed by S *provided* we can argue that SE outputs the $\mathsf{time\text{-}out}$ symbol with only negligible probability. In the claim below, we bound this probability. We remark that this is a standard claim, and similar claims have earlier appeared in several works before, e.g., [GK96, Lin01, Lin10].

Claim 2 *The simulator-extractor SE outputs the* $\mathsf{time\text{-}out}$ *symbol with only negligible probability.*

We now give proofs for Claim 1 and Claim 2 to complete the proof of Theorem 2.

Proof of Claim 1. Let T be a random variable that denotes the total running time of the algorithm. Further, let T_i be a random variable that denotes the running time of step (Si) in the above algorithm. Then, by definition, we have $T = T_1 + T_2 + T_3$. We now proceed to compute $E[T]$. Below, we will use the following fact for conditional expectation: given random variables X, Y,

$$E[X] = \sum_i E[X|Y = i].\Pr[Y = i] \tag{1}$$

First note that $E[T_1] = \mathrm{poly}(n)$ since S is an expected polynomial time algorithm. Let $\mathsf{S1}_{\mathsf{accept}}$ denote the event that the view output by S in step (S1) contains an accepting right execution.

[21]The effect of parallel execution can be obtained by running one step of the first procedure followed by one step of the second procedure, and so on.

Now note that if step (S2) is performed (which happens only if event $\mathsf{S1}_{\mathsf{accept}}$ occurs), the number of trials in (S2) are $\frac{n^2}{p_s}$ on the average. Then, we have $E[T_2|\mathsf{S1}_{\mathsf{accept}}] = \frac{\text{poly}(n)}{p_s}$. Note that $E[T_2|\neg\mathsf{S1}_{\mathsf{accept}}] = 0$. Then, using equation 1, we have:

$$\begin{aligned} E[T_2] &= E[T_2|\mathsf{S1}_{\mathsf{accept}}].\Pr[\mathsf{S1}_{\mathsf{accept}}] \\ &= \frac{\text{poly}(n)}{p_s} \cdot p_s = \text{poly}(n) \end{aligned}$$

Next, first note that $E[T_{3-a}|\neg\mathsf{S1}_{\mathsf{accept}}] = 0$. Then, using equation 1, we have:

$$\begin{aligned} E[T_{3-a}] &= E[T_{3-a}|\mathsf{S1}_{\mathsf{accept}}].\Pr[\mathsf{S1}_{\mathsf{accept}}] \\ &= 2^n \cdot p_s \end{aligned}$$

Now, recall that in step (S3-b-i), the adversary A^* is run at most $\frac{\text{poly}(n)}{\overline{p_s}}$ times. Since A^* is an (expected) polynomial time algorithm, we have, $E[T_{3-b-i}|\mathsf{S1}_{\mathsf{accept}}] = \frac{\text{poly}(n)}{\overline{p_s}}$. Note that $E[T_{3-b-i}|\neg\mathsf{S1}_{\mathsf{accept}}] = 0$. Then, using equation 1, we have:

$$\begin{aligned} E[T_{3-b-i}] &= E[T_{3-b-i}|\mathsf{S1}_{\mathsf{accept}}].\Pr[\mathsf{S1}_{\mathsf{accept}}] \\ &= \frac{\text{poly}(n)}{\overline{p_s}} \cdot p_s \end{aligned}$$

Now, let $\mathsf{S3}_{\mathsf{accept}}$ denote the event that an accepting view is obtained in step (S3-b-i). Recall that A^* produces an accepting view with probability p_{a^*}. From union bound, we have, $\Pr[\mathsf{S3}_{\mathsf{accept}}] = p_s \cdot (\frac{\text{poly(n)}}{\overline{p_s}} \cdot p_{a^*})$. Further, recall that the expected running time of the argument of knowledge extractor E is bounded by $\frac{\text{poly}(n)}{p_{a^*}}$ conditioned on the event that the input view is accepting. That is, $E[T_{3-b-ii}|\mathsf{S3}_{\mathsf{accept}}] = \frac{\text{poly}(n)}{p_{a^*}}$. Note that $E[T_{3-b-ii}|\neg\mathsf{S3}_{\mathsf{accept}}] = 0$. Then, using equation 1, we have,

$$\begin{aligned} E[T_{3-b-ii}] &= E[T_{3-b-ii}|\mathsf{S3}_{\mathsf{accept}}].\Pr[\mathsf{S3}_{\mathsf{accept}}] \\ &= \frac{\text{poly}(n)}{p_{a^*}} \cdot \left(p_s \cdot (\frac{\text{poly(n)}}{\overline{p_s}} \cdot p_{a^*})\right) = \frac{\text{poly}(n)}{\overline{p_s}} \cdot p_s \end{aligned}$$

From linearity of expectation, we have that $E[T_{3-b}] = E[T_{3-b-i}] + E[T_{3-b-ii}] = \frac{\text{poly}(n)}{\overline{p_s}} \cdot p_s$. Now, let $\mathsf{S2}_{\mathsf{const}}$ denote the event that the probability $\overline{p_s}$ estimated in step (S2) is within a constant factor of p_s. Then, we can consider the following two cases:

I. If event $\mathsf{S2}_{\mathsf{const}}$ occurs, then $E[T_{3-b}] = \text{poly}(n)$. In this case, step (S3-b) finishes earlier than step (S3-a). Thus, the total time contributed by step (S3) is $\text{poly}(n)$.

II. If event $\neg\mathsf{S2}_{\mathsf{const}}$ occurs, then step (S3-a) may finish earlier. In this case, the total time contributed by step (S3) is at most $2 \cdot (p_s \cdot 2^n)$.

Now note that event $\mathsf{S2}_{\mathsf{const}}$ occurs with (overwhelmingly high) probability $1 - 2^{-n^2}$. Therefore, using equation 1, we have:

$$\begin{aligned} E[T_3] &= E[T_3|\mathsf{S2}_{\mathsf{const}}].\Pr[\mathsf{S2}_{\mathsf{const}}] + E[T_3|\neg\mathsf{S2}_{\mathsf{const}}].\Pr[\neg\mathsf{S2}_{\mathsf{const}}] \\ &= \text{poly}(n) \cdot (1 - 2^{-n^2}) + p_s \cdot 2^{n+1} \cdot 2^{-n^2} = \text{poly}(n) \end{aligned}$$

Finally, by linearity of expectation, we have $E[T] = E[T_1] + E[T_2] + E[T_3] = \text{poly}(n)$. This completes the proof of Claim 1.

Proof of Claim 2. Let $\Delta(x, r)$ denote the probability that the simulator-extractor SE, on input theorem x and random tape r, outputs the time-out symbol. Then,

$$\begin{aligned}
\Delta(x,r) &= p_s \cdot \sum_{i \geq 1} \Pr(\lfloor 1/\overline{p_s} \rfloor = i) \cdot (1 - p_{a^*})^{i \cdot \mathrm{poly}(n)} \\
&< p_s \cdot \left(\Pr\left(p_s/\overline{p_s} = \Theta(1)\right) \cdot (1 - p_{a^*})^{\mathrm{poly}(n)/p_s} + \Pr\left(p_s/\overline{p_s} \neq \Theta(1)\right) \right) \\
&< p_s \cdot (1 - p_{a^*})^{\mathrm{poly}(n)/p_s}
\end{aligned}$$

We will now show that $\Delta(x, r)$ is negligible in n. Let us first assume to the contrary that that there exists a polynomial $P(\cdot)$, an infinite sequence of theorems $\{x_n\}$ (with $|x_n| = n$), and an infinite sequence of random tapes $\{r_n\}$, such that $\Delta(x_n, r_n) > 1/P(n)$. It is easy to see that for each such n, $p_s > 1/P(n)$. We now consider two cases:

Case 1: For infinitely many n's, $p_{a^*} > p_s/2$. In such a case, we have the following:

$$\begin{aligned}
\Delta(x_n, r_n) &\leq (1 - p_{a^*})^{\mathrm{poly}(n)/p_s} \\
&\leq (1 - p_s/2)^{\mathrm{poly}(n)/p_s} \\
&< 2^{-\mathrm{poly}(n)/2}
\end{aligned}$$

which is a contradiction.

Case 2: For infinitely many n's, $p_{a^*} < p_s/2$. Further, for all these n's, it also holds that $p_{a^*} \geq p_a - \mathrm{negl}(n)$ (for some negligible function $\mathrm{negl}(n)$). We consider the following two subcases:

1. $p_{a^*} \geq p_a$: In this case, we have that $p_a \leq p_s/2$. It follows that $p_s - p_a \geq p_s/2$, which is non-negligible since p_s is non-negligible in n. This contradicts the fact that p_s and p_a must be negligibly close.

2. $p_{a^*} < p_a$: In this case, there must exist a negligible function $\mathrm{negl}(n)$ such that $p_{a^*} = p_a - \mathrm{negl}(n)$. Then, once again we obtain $p_a \leq p_s/2$, and hence $p_s - p_a \geq p_s/2$. As in the case above, we have a contradiction.

Hence, contradiction follows in both cases. This completes the proof of Claim 2.

4.2 SimSound versus SimExt

We show that, perhaps somewhat surprisingly, a simulation extractable protocol is not necessarily simulation sound. Intuitively, this is because the behavior of the simulator-extractor SE is not defined for *false* statements. We exploit this fact to construct a protocol which is simulation extractable, but not simulation sound.

At a high level, we consider languages $L \in \mathcal{NP} \cap co\text{-}\mathcal{NP}$, since such languages admit polynomial-size witness for both types of assertions: "$x \in L$"and "$x \notin L$". We then take any SimExt protocol and modify it so that first the man-in-the-middle A proves to P that "$x \notin L$" using a tag-string $\mathtt{tag}$; if it succeeds, P proves that "$x \in L$" using a modified tag-string $\mathtt{tag}'$, which A can copy on right. If x is false, it is not hard to see that A succeeds if the left interaction succeeds in proving a false statement. We now give more details.

Protocol $\langle \hat{P}, \hat{V} \rangle$. Let $\langle P, V \rangle$ be a tag based simulation extractable interactive proof system for a language $L \in \mathcal{NP} \cap \textit{co-}\mathcal{NP}$. An interesting example of such a language L is quadratic-residuosity modulo a product of two primes. Thus, every statement x has a polynomial size witness w for either $x \in L$ or $x \notin L$ (i.e., $x \in \bar{L}$). Given $\langle P, V \rangle$, we construct a new system $\langle \hat{P}, \hat{V} \rangle$ (for L), such that $\langle \hat{P}, \hat{V} \rangle$ is simulation-extractable, but not simulation-sound. Protocol $\langle \hat{P}, \hat{V} \rangle$ is given in Figure 4.

Common Input: An instance x supposedly in L.
Private Input to $\hat{P}$: A witness $w \in R_L(x)$.
Parameters: Security parameter 1^n.
Tag String: `tag` of length n chosen by $\hat{P}$.

Stage 1:
$\hat{V} \Leftrightarrow \hat{P}$: $\hat{V}$ and $\hat{P}$ engage in an execution of $\langle P, V \rangle$ using the tag `tag`, where $\hat{V}$ proves to $\hat{P}$ that $x \notin L$. Note that the honest verifier always fails.

Stage 2:
$\hat{P} \Leftrightarrow \hat{V}$: $\hat{P}$ proves to $\hat{V}$ that $x \in L$ by engaging in an execution of $\langle P, V \rangle$ using the tag string $\texttt{tag}'$ where,

(a) $\texttt{tag}' = \texttt{tag}$ if $\hat{V}$ succeeds in stage 1,

(b) Otherwise, $\texttt{tag}'$ is the same as `tag` with its first bit flipped.

Figure 4: Protocol $\langle \hat{P}, \hat{V} \rangle$

We now briefly sketch the argument that $\langle \hat{P}, \hat{V} \rangle$ is simulation extractable, but not simulation-sound.

$\langle \hat{P}, \hat{V} \rangle$ is simulation extractable. Completeness and soundness of the protocol are easy to verify. In addition, consider the real experiment where A receives proof to a "true" statement $x \in L$ with tag string `tag`, and tries to prove statement $\tilde{x} \in L$ on the right. Since right verifier always aborts, it follows from the soundness of $\langle P, V \rangle$ that A fails as well in stage 1 of the left proof. This observation shows that it is easy to modify the simulator-extractor guaranteed for $\langle P, V \rangle$ so that it works for $\langle \hat{P}, \hat{V} \rangle$ as well.

$\langle \hat{P}, \hat{V} \rangle$ is not simulation-sound. Recall that in the man-in-the-middle experiment for simulation-soundness, the adversary A may chose a "false" statement x and obtain a *simulated* proof for $x \in L$ from the simulator in the left execution. The simulation-soundness property requires that even in such a scenario, A cannot prove (except with negligible probability) a false statement in the right execution as long as $\tilde{\texttt{tag}} \neq \texttt{tag}$. Let A be an adversary who has an $x \notin L$ along with a witness w for this fact, "hardwired" into it.[22] Let $\texttt{tag}'$ be the tag string `tag` with the first bit flipped. We will show that A can successfully prove that $x \in L$ (which is *false*) using tag-string $\tilde{\texttt{tag}} = \texttt{tag}'$ which is different from `tag`, violating the simulation-soundness.

[22] Note that since $L \in \mathcal{NP} \cap \textit{co-}\mathcal{NP}$, there exists a short witness for $x \notin L$.

A sends x to the simulator, receives string $\mathtt{tag}$, and sends $\tilde{\mathtt{tag}} = \mathtt{tag}'$ along with $\tilde{x} = x$ to the right-hand-side verifier. On right-hand-side honest V fails.[23] However, A uses its hardwired witness to succeed in stage 1 on left. As a result, S now must convince A that $x \in L$ using the tag string $\mathtt{tag}'$ in stage 2. Now A simply "copies" the left proof entirely to stage 2 of the right execution. (Note that since the honest verifier must have failed to prove that $x \notin L$ in Stage 1, A is expected to prove that $x \in L$ using tag-string $\mathtt{tag}'$ in Stage 2 of the right execution.) That is, A simply acts as a wire and forwards each message from the simulator in the left execution to the honest verifier in the right execution; each response from the verifier is forwarded back to the simulator. Since the simulator succeeds in convincing A in the left execution, A succeeds as well in the right execution.

References

[Bar02] Boaz Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *FOCS*, 2002.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, October 1988.

[BCFW09] Alexandra Boldyreva, David Cash, Marc Fischlin, and Bogdan Warinschi. Foundations of non-malleable hash and one-way functions. In *ASIACRYPT*, pages 524–541, 2009.

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO*, pages 26–45, 1998.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC*, pages 103–112, 1988.

[BFO08] Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In *Advances in Cryptology – CRYPTO '08*, pages 335–359, 2008.

[BFOR08] Mihir Bellare, Marc Fischlin, Adam O'Neill, and Thomas Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In *Advances in Cryptology – CRYPTO '08*, pages 360–378, 2008.

[Blu87] Manuel Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, pages 1444–1451, 1987.

[BNP08] Assaf Ben-David, Noam Nisan, and Benny Pinkas. Fairplaymp: a system for secure multi-party computation. In *ACM Conference on Computer and Communications Security*, pages 257–266, 2008.

[BSMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM J. Comput.*, 20(6):1084–1118, 1991.

[23] If simulator-extractor somehow convinces A on right in stage-1, A will change its strategy on left and fail. This works as well.

[CKOS01] Giovanni Di Crescenzo, Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Efficient and non-interactive non-malleable commitment. In *EUROCRYPT*, pages 40–59, 2001.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party computation. In *Proc.* 34*th STOC*, pages 494–503, 2002.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC*, pages 151–158, 1971.

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

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437 (electronic), 2000. Preliminary version in STOC 1991.

[DDO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In *CRYPTO ' 2001*, pages 566–598, 2001.

[DPW10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In *ICS*, pages 434–452, 2010.

[DW09] Yevgeniy Dodis and Daniel Wichs. Non-malleable extractors and symmetric key cryptography from weak secrets. In *STOC*, pages 601–610, 2009.

[FS89] U. Feige and A. Shamir. Zero knowledge proofs of knowledge in two rounds. In *CRYPTO*, pages 526–545, 1989.

[FS90] U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *Proc.* 22*nd STOC*, pages 416–426, 1990.

[GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–189, Summer 1996.

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *Proc.* 17*th STOC*, pages 291–304, 1985.

[GMY06] Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. *J. Cryptology*, 19(2):169–209, 2006.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001. Earlier version available on `http://www.wisdom.weizmann.ac.il/~oded/frag.html` .

[Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In *STOC*, pages 695–704, 2011. See full version available on `http://research.microsoft.com/en-us/people/vipul/nmcom.pdf`.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.

[KY00] Jonathan Katz and Moti Yung. Complete characterization of security notions for probabilistic private-key encryption. In *STOC*, pages 245–254, 2000.

[Lev84] Leonid A. Levin. Problems, complete in "average" instance. In *STOC*, page 465, 1984.

[Lin01] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. In *Crypto '01*, pages 171–189, 2001.

[Lin10] Yehuda Lindell. Constant round zero knowledge proofs of knowledge. 2010. `http://eprint.iacr.org/2010/487.pdf`.

[Lin11] Yehuda Lindell. Highly-efficient universally-composable commitments based on the ddh assumption. In *EUROCRYPT*, pages 446–466, 2011.

[LP09] Huijia Lin and Rafael Pass. Non-malleability amplification. In *STOC*, pages 189–198, 2009.

[LP11] Huijia Lin and Rafael Pass. Constant-round non-malleable commitments from any one-way function. In *STOC*, 2011.

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

[LPV09] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. A unified framework for concurrent security: universal composability from stand-alone non-malleability. In *STOC*, pages 179–188, 2009.

[MNPS04] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay - secure two-party computation system. In *USENIX Security Symposium*, pages 287–302, 2004.

[Nao89] Moni Naor. Bit commitment using pseudo-randomness (extended abstract). In *CRYPTO*, pages 128–136, 1989.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, pages 427–437, 1990.

[OPV10] Rafail Ostrovsky, Omkant Pandey, and Ivan Visconti. Efficiency preserving transformations for concurrent non-malleable zero knowledge. In *TCC*, pages 535–552, 2010.

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *Proc.* 36*th STOC*, pages 232–241, 2004.

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

[PR05a] Rafael Pass and Alon Rosen. Concurrent non-malleable commitments. In *FOCS*, 2005.

[PR05b] Rafael Pass and Alon Rosen. New and improved constructions of non-malleable cryptographic protocols. In *STOC*, 2005.

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

[PW09] Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In *TCC*, pages 403–418, 2009.

[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] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *Proc. 40th FOCS*, pages 543–553, 1999.

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

Appendix

A A Note on using Goyal's Scheme

To obtain constant round result, the non-malleable commitment scheme $\langle C, R\rangle$ in our construction should be replaced by a simple variant of Goyal's scheme, denoted by $\langle C_g, R_g\rangle$. Let us explain: denote by NMCOM_g the black-box version of Goyal's scheme (see appendix B, page 20, of [Goy11]). This scheme does not satisfy definition 1, which is "non-malleability w.r.t. commitment". It satisfies the following weaker notion.

Non-malleability w.r.t. Replacement. Recall that the non-malleability definition provides A with value $\tilde{v}$ during the construction of view $\mathsf{mim}^A_{\langle C,R\rangle}(v_0, z, \mathtt{tag})$. The replacement definition considers a *replacer* (not necessary PPT) which does the following: when A is supposed to get the value $\tilde{v}$, the replacer instead gives him a different value $\tilde{rv}$ satisfying the following. If $\tilde{v} \neq \perp$, then except with probability at most p (over the entire experiment), the two values are set equal: $\tilde{rv} = \tilde{v}$. However, if $\tilde{v} = \perp$, replacer might set $\tilde{rv}$ to any arbitrary string. Let $\mathsf{rmim}^A_{\langle C,R\rangle}(v_b, z, \mathtt{tag})$ denote A's view in such an experiment for $b \in \{0,1\}$. The scheme is said to be "non-malleable w.r.t. replacement", if $\mathsf{rmim}^A_{\langle C,R\rangle}(v_0, z, \mathtt{tag}) = \mathsf{rmim}^A_{\langle C,R\rangle}(v_1, z, \mathtt{tag})$ for all v_0, v_1 and $p = 1/\text{poly}(n)$.

It is also shown, that if the length of the tags is $\log n + 1$, then NMCOM_g is also "one-many non-malleability w.r.t. replacement for tags of length $\log n + 1$". Recall that in one-many experiment, A interacts with many receivers $R_1, \ldots, R_m$ (and not just one), while still interacting with only one committer C on left. When A finishes *all* his commitments on right, the values committed by him—say $\tilde{v}_1, \ldots, \tilde{v}_m$—are given to A. In case of replacement, A instead receives $\tilde{rv}_1, \ldots, \tilde{rv}_m$. Other than this change, everything else remains the same and we require indistinguishability between the two views.

Scheme $\langle C_g, R_g\rangle$. Given a tag $\mathtt{tag} \in \{0,1\}^n$, apply the "DDN-LOGN Trick" to obtain n small tags $\mathtt{tag}_1, \ldots, \mathtt{tag}_n$, each of length $\log n + 1$: $\mathtt{tag}_i := i \circ \mathtt{tag}[i]$. To commit to a string $v \in \{0,1\}^n$, the scheme $\langle C_g, R_g\rangle$ simply runs n parallel execution of NMCOM_g one for each small tag $\mathtt{tag}_i$ (which commits to the bit $v[i]$.

Any change in the proof? We only need to make a very minor change, so that instead of reducing the proof to "non-malleability w.r.t. commitment", it will be reduced to "one-many non-malleability w.r.t. replacement for tags of length $\log n + 1$". And to do this, recall that in the proof, V's commitment is obtained from an external committer C. When using $\langle C_g, R_g \rangle$, instead of receiving the whole commitment from outside, we will continue to simulate C_g inside except that the commitment for small tag $\mathtt{tag}_j$ will be received from outside (on some bit $v[j]$). The index j will be the one for which $\tilde{\mathtt{tag}}_j \neq \mathtt{tag}_i$ for every $i \in [n]$ (and such a j always exists, with ties broken arbitrarily). The proof remains identical otherwise.

B Other Standard Definitions

B.1 Extractor for $\langle \widehat{C}_s, \widehat{R}_s \rangle$

For an arbitrary probabilistic polynomial time committing algorithm C^*, define $\mathtt{trans}_{\langle C^*, \widehat{R}_s \rangle}$ to be the following random variable: fix uniformly random coins for C^* and $\widehat{R}_s$, and output the *transcript* of interaction—i.e., all messages exchanged—between C^* and $\widehat{R}_s$. Let $\mathsf{Value}(\mathtt{trans}_{\langle C^*, \widehat{R}_s \rangle})$ denote the string committed to by C^* in this interaction.

Claim 3 *For every probabilistic polynomial time Turing machine C^*, there exists a probabilistic Turing machine $CExt$ such that for every polynomial $q(\cdot)$ and every sufficiently large $n \in \mathbb{N}$ the following conditions hold:*

1. *For ν sampled according to $\mathtt{trans}_{\langle C^*, \widehat{R}_s \rangle}$,*

$$\Pr_{\nu}\left[\mathsf{Value}(\nu) \neq \perp \wedge \mathtt{str} \leftarrow CExt^{C^*}(\nu) \wedge \mathtt{str} \neq \mathsf{Value}(\nu)\right] < \frac{1}{q(n)}$$

2. *The expected running time of $CExt$ is bounded by,*

$$\frac{\mathrm{poly}(n)}{\Pr\left[R_s \text{ accepts } \mathtt{trans}_{\langle C^*, \widehat{R}_s \rangle}\right]}$$

The proof for the above claim follows from [Ros04] and is therefore omitted.

B.2 Zero Knowledge, Interactive Proofs, Proofs of Knowledge

Here we recall the standard definitions of interactive proofs, zero knowledge [GMR85], and proofs of knowledge [GMR85, FS89]. For convenience, we will follow the notation and presentation of [PR05b]. Let P (called the *prover*) and V (called the *verifier*) denote a pair of probabilistic interactive Turing machines that are running a protocol with each other on common input x. Throughout our text, we will always assume V to be a polynomial-time machine. Let $\langle P, V \rangle(x)$ be the random variable representing the output of V at the end of the protocol. If the machine P is polynomial-time, it is assumed that it has a private input w.

Definition 5 (Interactive proof system) *A pair of probabilistic interactive Turing machines $\langle P, V \rangle$ is called an* interactive proof system *for a language L and a witness relation R_L if the following two conditions hold:*

1. Completeness: *For every $x \in L$,*

$$\Pr[\langle P, V\rangle(x) = 1] \geq 1 - \text{negl}(|x|)$$

2. Soundness: *For every $x \notin L$, and every interactive Turing machine P^*,*

$$\Pr[\langle P^*, V\rangle(x) = 1] \leq \text{negl}(|x|)$$

Here, probability is taken over the coins of all the interactive Turing machines.

Zero Knowledge. An interactive proof $\langle P, V\rangle$ is said to be *zero-knowledge* if, informally speaking, the verifier V learns nothing beyond the validity of the statement being proved. This intuition is formalized by requiring that the view of every probabilistic polynomial-time (PPT) verifier V^*, represented by $\text{view}_{V^*}(x, z)$, generated as a result of its interaction with P can be "simulated" by a PPT machine S (referred to as the *simulator*). Here, the verifier's view consists of the common input x, its random tape, and the sequence of prover messages that it receives during the protocol execution. The auxiliary input of V^* and S is denoted by $z \in \{0,1\}^*$.

Definition 6 (Zero knowledge) *An interactive proof system $\langle P, V\rangle$ is said to be* zero knowledge *if for every* PPT *machine V^*, there exists a* PPT *algorithm S such that for every $x \in L$, every $z \in \{0,1\}^*$, $\text{view}_{V^*}(x, z)$ and $S(x, z)$ are computationally indistinguishable.*

One can consider stronger variants of zero knowledge where the output of S is statistically close (or identical) to the verifier's view. In this paper, we will focus on the computational variant only.

Proofs of Knowledge. An interactive proof system is a proof of knowledge if, informally speaking, not only does the prover convince the verifier of the validity of the statement, but it also possesses a witness for the statement. This intuition is formalized by showing the existence of a machine E, called the *knowledge extractor*, that is able to extract a witness from a prover that succeeds in convincing an honest verifier. We use the following variant of the definition of proof of knowledge, presented in [PR05b].

Definition 7 (Proof of knowledge) *Let $\langle P, V\rangle$ be an interactive proof system for the language L with witness relation R_L. We say that (P, V) is a* proof of knowledge *if there exists a polynomial $q(\cdot)$ and a probabilistic oracle machine E, such that for every computationally unbounded machine P^*, for every $x \in L$, and every $y, r \in \{0,1\}^*$, the following properties hold:*

1. The expected number of steps taken by E is bounded by

$$\frac{q(|x|)}{\Pr_{r'}[\langle P^*_{x,w,r}, V(x; r')\rangle = 1]}$$

2. The machine E with oracle access to $P^*_{x,w,r}$ outputs a solution $w^* \in R_L(x)$ with probability at least $1 - \text{negl}(|x|)$.

Here $P^*_{x,w,r}$ denotes the deterministic machine P^* with common input fixed to x, auxiliary input fixed to w, and random tape fixed to r. The machine E is called a knowledge extractor.