Constant-Round Coin-Tossing With a Man in the Middle or Realizing the Shared Random String Model

Boaz Barak*

May 19, 2008

Abstract

We construct the first *constant-round* non-malleable commitment scheme and the first *constant-round* non-malleable zero-knowledge argument system, as defined by Dolev, Dwork and Naor. Previous constructions either used a non-constant number of rounds, or were only secure under stronger setup assumptions. An example of such an assumption is the *shared random string model* where we assume all parties have access to a reference string that was chosen uniformly at random by a trusted dealer.

We obtain these results by defining an adequate notion of *non-malleable coin-tossing*, and presenting a *constant-round* protocol that satisfies it. This protocol allows us to transform protocols that are non-malleable in (a modified notion of) the *shared random string model* into protocols that are non-malleable in the *plain* model (without any trusted dealer or setup assumptions). Observing that known constructions of a non-interactive non-malleable zero-knowledge argument systems in the shared random string model are in fact non-malleable in the modified model, and combining them with our coin-tossing protocol we obtain the results mentioned above.

The techniques we use are different from those used in previous constructions of non-malleable protocols. In particular our protocol uses diagonalization and a *non-black-box* proof of security (in a sense similar to Barak's zero-knowledge argument).

Keywords: two-party protocols, man in the middle setting, malleability, coin-tossing protocols, black-box vs. non-black-box simulation, non-malleable commitments, non-malleable zero-knowledge, universal arguments, CS proofs

*Department of Computer Science, Weizmann Institute of Science, Israel. Email: `boaz@wisdom.weizmann.ac.il`

Contents

List of Figures

List of Tables

1 Introduction

1.1 Overview

In the *man-in-the-middle* (MIM) attack on a cryptographic two-party protocol, the adversary has complete control over the communication channel between two honest parties. The adversary has the power not only to read all messages sent between the parties, but also actively change, erase or insert its own messages into the channel. This attack can be very relevant in many practical settings, and thus designing protocol secure against such an attack is an important task.

Dolev, Dwork and Naor [DDN91] considered the MIM attack and defined a protocol to be *non-malleable* if it remains secure under such attack.[1] In the *plain* model (where there are no setup assumptions such as a public key infrastructure or a shared random string), [DDN91] constructed non-malleable protocols for the fundamental cryptographic tasks of commitment and zero-knowledge. However, the protocols of [DDN91] took a *non-constant* number of communication rounds (logarithmic in the security parameter). In this work, we focus on the task of constructing *constant-round* protocols for these tasks.

The shared random string model. Unlike the case in the plain model, in the *shared random string model*, introduced by Blum, Feldman and Micali [BFM88], one assumes that there exists a trusted party that chooses a string uniformly at random and sends it to all parties (including the adversary) before the protocol is executed. This setup assumption enables the construction of much more round efficient protocols. In fact, Sahai [Sah99] constructed a 1-round (i.e., *non-interactive*) non-malleable zero-knowledge argument for **NP** in this model, whereas Di Crescenzo, Ishai and Ostrovsky [DIO98] constructed a 1-round non-malleable commitment scheme in this model (see Section 1.3 for details on other constructions and related works in this model).

Our results and techniques. In this paper we give the first construction of *constant-round* non-malleable commitment and zero-knowledge schemes in the *plain* model, without any setup assumption. Our approach is to first construct a non-malleable *coin-tossing* protocol. We then use this coin-tossing protocol in order to transform a non-malleable protocol (such as a zero-knowledge proof or a commitment scheme) from the shared random string model into the plain model.

The techniques utilized in the construction of the non-malleable coin-tossing protocol are different from those used in previous works in non-malleable cryptography. In particular our proof of security involves a diagonalization argument and a non-black-box use of the code of the adversary's algorithm. Similar techniques were first used in [Bar01] in the context of zero-knowledge systems. This works demonstrates that these techniques are applicable also in other settings in cryptography.

Erratum. Since the publication of the proceedings version of this paper, I have discovered an error in the proof of the commitment scheme for non-uniform adversaries. (This is an extension of the proof for non-uniform adversaries that was claimed in the proceedings version but was not included there for lack of space.) The Theorem however is still correct, and the proof can be fixed by making an appropriate change in the definition and usage of "evasive set families" (see Section 4.1). We note that between the time of the publication of the conference version of this paper (Oct 2002) and the discovery of the bug and fix (March 2005), Pass and Rosen [PR05] came up with an alternative

[1]The definition of non-malleability is usually described in somewhat different terms. See also Remark 1.1

$$\begin{tikzcd} \boxed{\text{L}} \arrow[r, shift left=3] \arrow[r, shift right=3] & \boxed{\text{C}} \arrow[l] \arrow[r, shift left=3] \arrow[r, shift right=3] & \boxed{\text{R}} \arrow[l] \end{tikzcd}$$

Left session　　　Right session

Figure 1: The MIM setting: all communication between L and R is done through the adversary C.

construction of non-malleable commitments (also using non-black-box techniques) whose analysis works in both the uniform and non-uniform setting. Their construction also has the advantage of having a considerably simpler analysis, that also uses quantitatively weaker assumptions.

1.2 Model and Basic Terminology

In this work we are only interested in two-party protocols. We will denote the two parties by the letters L and R (L stands for left, R for right). Two examples that are worth keeping in mind are *commitment schemes* and *zero-knowledge proofs*. In a *commitment scheme* the left player L is the *sender* that wants to commit to a value, while the right player R is the *receiver* that receives the committed value. In a *zero-knowledge proof* the left player L is the *prover* that wants to convince the right player R (called the *verifier*) that some statement is true.

In the *man-in-the-middle setting* (MIM), as depicted in Figure 1, there is a third party denoted by C (C can stand for either *center* or *channel*, we will typically call C the *adversary*). All the communication between L and R is done through C. Thus, both players L and R only talk to C and cannot communicate directly with each other. The adversary C can decide to simply relay the messages each party sends to the other party, but it can also decide to block, delay, or change messages arbitrarily. Thus, if L and R wish to run a two-party protocol Π in the MIM setting, then we can think of the protocol Π as being executed in *two concurrent* sessions. In one session L plays the left side and the adversary C plays the right side, and in the second session C plays the left side and R plays the right side. We assume that the adversary C controls the scheduling of messages in both sessions. We call the first session (where L interacts with C) the *left* session, and the second session (where C interacts with R) the *right* session.

1.2.1 Unavoidable strategies.

There are two strategies that the adversary C can always use without being detected. One strategy is the *relaying* strategy in which the only thing C does is relay the messages between L and R. In this case C is transparent and this is equivalent to a single execution of the protocol Π between L and R. The other unavoidable strategy is the *blocking* strategy in which C plays its part in each session completely independent of the other session. In each session, C uses the honest strategy to play its part (i.e., in the left session C uses the strategy of the honest right player and in the right session C uses the strategy of the honest left player.) Clearly, regardless of the protocol Π, it is impossible to prevent the adversary from using one of these two strategies. Therefore, we call the relaying and blocking strategies the *unavoidable* strategies. Intuitively, the goal in designing protocols for the man-in-the-middle setting, is to design protocols that force C to use one of the two

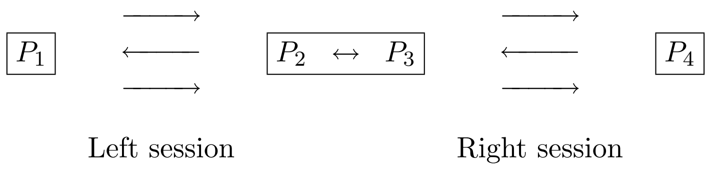


Figure 2: Non-malleability, as defined by [DDN91] : P_2 and P_3 are controlled by the adversary

unavoidable strategies (or such that it could not be advantageous to C to use any other strategy).[2]

For example, consider the case of a commitment scheme. When executed in the man-in-the-middle setting, in the left session the player L commits to a value α to C that plays the receiver, whereas in the right session C plays the sender and commits to a value $\widetilde{\alpha}$.[3] If C uses the *relaying* strategy then it holds that $\alpha = \widetilde{\alpha}$. On the other hand, if C uses the *blocking* strategy then it holds that $\widetilde{\alpha}$ is independent of α. Indeed, loosely speaking, the goal in *non-malleable* commitments is to design a commitment scheme such that regardless of the strategy C uses, it will hold that either $\widetilde{\alpha}$ is equal to α or that $\widetilde{\alpha}$ is independent of α.[4]

Remark 1.1 (Comparison with [DDN91])**.** Dolev *et al.*[DDN91] used different notations to describe essentially the same setting. They considered four parties P_1, P_2, P_3, P_4 that execute a two-party protocol in two concurrent sessions (see Figure 2). The left session is between P_1 and P_2, and the right session is between P_3 and P_4. Both P_2 and P_3 are controlled by an adversary, whereas P_1 and P_4 are honest and follow the protocol (and thus, P_1 is oblivious to the (P_3, P_4) interaction and P_4 is oblivious to the (P_1, P_2) interaction). This means that P_2 and P_3 combined correspond to the adversary C in our notation, and that P_1 corresponds to L in our notation, where P_4 corresponds to R in our notation. Previous works also used somewhat different emphasis in presenting the goal of non-malleable protocols. For example, the goal of a non-malleable commitment scheme is usually described as to ensure that the committed values in both sessions are *independent* (i.e., that the adversary is using the *blocking* strategy). The possibility of the values being *identical* (i.e., that the adversary will use the *relaying* strategy) is also allowed because it is unavoidable, but it is considered to be an uninteresting special case. In contrast, we treat both strategies equally. Note also that in this work we only consider protocols that are non-malleable with respect to *themselves* (as defined by [DDN91]), where [DDN91] defined also non-malleability with respect to general protocols.

1.2.2 Scheduling strategies

The adversary in the MIM model can control not only *what* is written in the messages sent but also decide *when* to send them. That is, the adversary has complete control over the scheduling of the messages. An important example of a scheduling strategy is the *synchronizing* scheduling. In this scheduling, the adversary synchronizes the two executions by immediately sending the i^{th}

[2]Actually, the adversary can always use also a "mixed" strategy in which it follows the relaying strategy with probability p, and the blocking strategy with probability $1 - p$, for some $p \in [0, 1]$.

[3]We only consider *statistically binding* commitment schemes, and so the committed value is determined uniquely by the transcript of the session.

[4]Actually, one needs to define an appropriate notion of "computational independence", as is done in [DDN91]. See also Section 5.

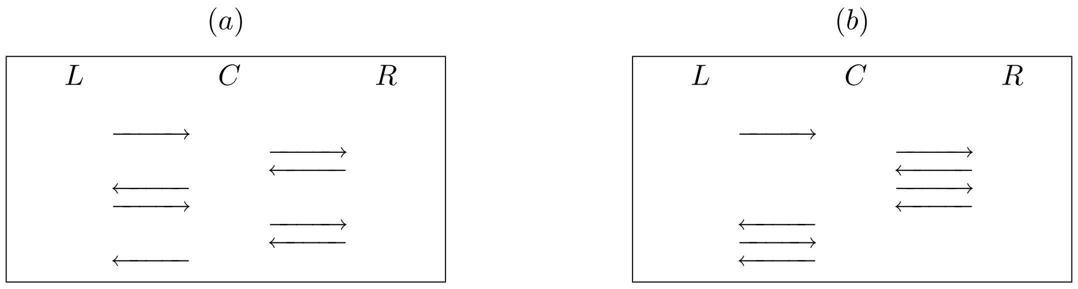


Figure 3: Two different scheduling strategies: (a) The man in the middle applies the "synchronizing" strategy. (b) The man in the middle does not wait for L's reply before answering R's message.

message in the right session after it receives the i^{th} message in the left session and vice versa (i.e., it sends the j^{th} message in the left session immediately after it received the j^{th} message in the right session). We call an adversary that always uses this scheduling a *synchronizing* adversary. We call an adversary that uses a different scheduling a *non-synchronizing* adversary. The difference between a synchronizing and a non-synchronizing adversary is illustrated in Figure 3. Note that even when restricted to the synchronizing scheduling, it is still possible for the adversary to use either the blocking or the relaying strategies. Thus, these strategies remain unavoidable even when restricting to synchronizing adversaries.

Unlike other settings in cryptography, such as concurrent zero-knowledge [DNS98, RK99, CKPR01, PRS02], in our setting the ability to control the scheduling does *not* add much power to the adversary. In fact, as we will show in Section 6, it is possible to transform any non-malleable commitment or zero-knowledge scheme that is secure against *synchronizing* adversaries, into a scheme secure against *general* (possibly non-synchronizing) adversary. Therefore, in most of this paper we will restrict ourselves into dealing only with synchronizing adversaries.[5] As mentioned above, in Section 6 we will show how to overcome this restriction.

1.3 The Shared Random String Model

In the *shared random string model* [BFM88], we assume the existence of a third trusted party called the *dealer*. Before a protocol is executed in this model, the dealer picks a string r uniformly at random and sends it to all the parties. The string r is called the *reference string*, and is given as an additional public input to the protocol. In the setting we are interested in (the MIM setting for a two-party protocol), this means that in a preliminary phase, the dealer sends the string r to L, R and C. After this phase, the protocol is executed in both the left and right sessions, with r as the public reference string.

Previous works. As mentioned above, unlike the case in the plain model, there are several round-efficient non-malleable protocols known in the shared random string model. Sahai [Sah99]

[5]In some sense, the synchronizing scheduling is the hardest schedule to deal with when designing a non-malleable protocol. In fact, one intuition behind the non-malleable commitment protocol of [DDN91] (and also an earlier work in a different model by Chor and Rabin [CR87]) is that the protocol is designed to force the adversary to use a *non-synchronizing* strategy.

constructed a single-round (i.e., *non-interactive*) non-malleable zero-knowledge proof system in this model. This scheme was improved by De Santis, Di Crescenzo, Ostrovski, Persiano and Sahai [DDO$^+$01]. Di Crescenzo, Ishai and Ostrovsky [DIO98] constructed in the shared random string model a non-interactive commitment scheme that is non-malleable in a weaker sense than [DDN91] ("non-malleable w.r.t. opening" [FF00]). Di Crescenzo, Katz, Ostrovski, and Smith [DKOS01, Sec. 3] constructed in the shared random string model[6] a non-interactive commitment satisfying the stronger notion of non-malleability (i.e., "non-malleable w.r.t. committing") defined in [DDN91]. Canetti and Fischlin [CF01] constructed in the shared random string model[7] non-interactive *universally composable* commitments which is a stronger notion than non-malleability. Interestingly, it is *impossible* to construct universally composable commitments in the plain model [CF01].

1.4 Non-malleable Coin-Tossing

The goal of this paper is to convert two-party protocols that are secure against a MIM attack in the *shared random string* model, into protocols that are secure against such an attack in the *plain* model. Toward this end we will want to construct a *non-malleable coin-tossing protocol* (in the plain model). Once we have such a coin-tossing protocol, we will convert a two-party protocol Π_{Ref} with a reference string that is secure in the shared random string model into a protocol Π_{Plain} in the plain model in the following way:

Construction 1.2 (Composition of a coin-tossing protocol with a reference string protocol)**.**

Phase 1: Run the coin-tossing protocol. Let r denote the result of this execution.

Phase 2: Run the protocol Π_{Ref} using r as the common reference string.

Loosely speaking, our goal is to construct a coin-tossing protocol such that whenever Π_{Ref} was secure in the shared random string model, the protocol Π_{Plain} will be secure in the plain man-in-the-middle model (with no shared string).

1.4.1 The modified shared random string model

Suppose that we execute the protocol Π_{Plain}, as constructed in Construction 1.2, in the man-in-the-middle setting. Let r denote the result of Phase 1 (the coin-tossing protocol) in the left session and let $\tilde{r}$ denote the result of Phase 1 in the right session. If we wish to emulate exactly the shared random string model then we want to ensure that $r = \tilde{r}$. Indeed, this will be the case if C will act transparently (i.e., use the *relaying* strategy, as described in Section 1.2.1). However, we have no guarantee that C will indeed use this strategy. In fact, C can always ensure that $\tilde{r}$ is independent of r, by using the *blocking* strategy.

The above problem motivates us in defining (for the MIM setting) a new ideal model called the *modified shared random string model.* In this model, the trusted dealer generates *two* random strings $r^{(1)}$ and $r^{(2)}$ uniformly and independently of one another. Then $r^{(1)}$ is used in the left

[6] Their construction is in the common reference string (CRS) model which is a slight generalization of the shared random string model. However they remark that under standard assumptions (e.g., hardness of factoring), their construction can be implemented in the shared random string model.

[7] Footnote 6 (Yehuda Lindell, personal communication, April 2002).

session (between L and C), but the adversary C is allowed to choose whether it wants to use the same string $r^{(1)}$ also in the right session (between C and R), or whether it wants to use the new independent string $r^{(2)}$. We allow the adversary to view the two strings before it makes this decision.[8]

Intuitively, it seems that the ability to choose that the strings used in both sessions will be independent should not help the adversary. Rather, it will only make the information that the adversary receives in the left session useless in the right session, and vice versa. Indeed, it turns out that many known protocols that are secure in the shared random string model, are also secure in the *modified* shared random string model. Thus we set our goal to constructing a coin-tossing protocol that would allow us to convert any protocol Π_{Ref} secure in the *modified* shared random string model into a protocol Π_{Plain} secure in the plain MIM setting. We provide an adequate definition, which we call *non-malleable coin-tossing protocol*, that indeed achieves this goal.

1.4.2 Definition of non-malleable coin-tossing

A *non-malleable coin-tossing* protocol is a protocol that implements the ideal functionality of the modified shared random string model, in the real (or plain) model, where there is no trusted third party. The formal definition is as follows:

Definition 1.3 (Non-malleable coin-tossing)**.** Let $\Pi = (L, R)$ be a (plain) two-party protocol. We say that Π is a non-malleable coin-tossing protocol if the following holds. For any efficient algorithm C there exists an efficient algorithm $\widehat{C}$ such that the following random variables are computationally indistinguishable:

1. $\mathsf{output}_{(L,R,C),\Pi}(1^n)$ where this denotes the triplet of outputs of L,R and C when executing Π in two concurrent sessions.

2. $(r^{(1)}, r^{(b)}, \tau)$ where this triplet is generated by the following experiment: first $r^{(1)}, r^{(2)}$ are chosen uniformly and independently in $\{0,1\}^n$. Then we let $(b, \tau) \leftarrow \widehat{C}(r^{(1)}, r^{(2)})$.

Definition 1.3 follows the paradigm of simulating an adversary C in the real model (i.e., the plain MIM setting) by an ideal adversary $\widehat{C}$ in the ideal model (i.e., the modified shared random string model). Indeed, Item 1 corresponds to the output of all parties in when the coin-tossing protocol is executed in the plain MIM model with adversary C, while Item 2 is the output of all parties when they interact with the trusted dealer of the modified shared random string model with adversary $\widehat{C}$.

1.5 Our Results

Our main result is the following:

Theorem 1.4. *Suppose that there exist hash functions that are collision-resistent against 2^{n^ϵ}-sized circuits for some $\epsilon > 0$. Then, there exists a constant-round non-malleable coin-tossing protocol.*

[8] We do not know whether or not it is unavoidable to allow the adversary to view both strings or at least one of them, if we want a model that can be simulated in the plain MIM setting.

By following Construction 1.2, and composing the coin-tossing protocol of Theorem 1.4 with a non-malleable protocol in the shared random string model, such as the non-interactive zero-knowledge of [DDO$^+$01],[9] we obtain the following theorem:

Theorem 1.5. *Suppose that there exist trapdoor permutations and collision-resistent hash functions strong against 2^{n^ϵ}-sized circuits for some $\epsilon > 0$. Then:*

1. *There exists a constant-round non-malleable zero-knowledge argument system for* **NP**.
2. *There exists a constant-round non-malleable (statistically binding) commitment scheme.*

Remarks. We note that our result can be obtained also somewhat weaker complexity assumptions. Note also that the non-malleable commitment scheme we obtain is non-malleable with respect to committing non-malleable *with respect to committing* (as the definition of [DDN91], which stronger than the definition of [DIO98], see [FF00]). We also note that, like the previous protocols of [DDN91], our schemes are *liberal* non-malleable in the sense that our simulator runs in *expected* polynomial-time. However, we believe that one can obtain the stronger notions using the techniques of Barak and Lindell [BL02]. See Section 7 for more discussion and details.

General theorems. It would have been nice if we proved two general statements of the form (**1**) "every protocol that is secure in the shared random string model is also secure in the *modified* shared random string model" and (**2**) "for every protocol that is secure in the modified shared random string model, its composition using Construction 1.2 with a non-malleable coin-tossing protocol yields a protocol that is secure in plain MIM setting". However this is problematic, not so much because Definition 1.3 is too weak, but mainly because the notion of "security" for protocols is not well-defined and depends on the particular application. We do however prove more general statements than Theorem 1.5: see Section 5 for more details.

1.6 Organization

Section 2 contains some notations, and the cryptographic primitives and assumptions that we use. In Section 3 we construct a non-malleable coin-tossing protocol that is secure against *uniform* polynomial-time adversaries. In Section 4 we show how to modify the construction to obtain security against *non-uniform* adversaries. In Section 5 we show how we can use our non-malleable coin-tossing protocol to obtain a non-malleable zero-knowledge and commitment schemes. In Section 6 we show how we can convert a non-malleable zero-knowledge or commitment scheme that is secure against adversaries that use the *synchronizing* scheduling, into a scheme that is secure general adversaries, that may use different scheduling strategies. Section 7 contains some remarks on the constructions and open questions.

We note that if one wants just to "get a taste" of our techniques and constructions, it is possible to read just Sections 3.1 and 3.2 to see a simple construction of a simulation sound zero-knowledge system secure against uniform adversaries. Simulation soundness is an important relaxation of non-malleability, and this construction illustrates some of the ideas used in the other sections.

[9] Actually, we will use a variation of this protocol, which will be interactive (but constant-round), in order to avoid assuming the existence of dense cryptosystems.

2 Preliminaries

2.1 Notation

For a finite set $S \subseteq \{0,1\}^*$, we write $x \leftarrow_{\mathrm{R}} S$ to say that x is distributed uniformly over the set S. We denote by U_n the uniform distribution over the set $\{0,1\}^n$. In all our protocols, we will denote the security parameter by n. A function $\mu(\cdot)$ from non-negative integers to reals is called *negligible* if for every constant $c > 0$ and all sufficiently large n, it holds that $\mu(n) < n^{-c}$. We say that an event happens with *overwhelming* probability if it happens with probability $1 - \mu(n)$ for some negligible function $\mu(\cdot)$. We will sometimes use neg to denote an unspecified negligible function.

For two strings α and β, we denote the concatenation of α and β by $\alpha \circ \beta$. If C is an algorithm then we denote by $\mathsf{desc}(C)$ the description of the code of the algorithm.

Computational indistinguishability. Let X and Y be random variables over $\{0,1\}^n$ and let $s \geq n$. We say that X and Y are *indistinguishable by s-sized circuits* if for every circuit D of size s, it holds that $|\Pr[D(X) = 1] - |\Pr[D(Y) = 1]| < \frac{1}{s}$. A *probability ensemble* is a sequence $\{X_i\}_{i \in I}$ of random variables, where I is an infinite subset of $\{0,1\}^*$ and X_i ranges over $\{0,1\}^{p(|i|)}$ for some polynomial $p(\cdot)$. We say that two probability ensembles $\{X_i\}_{i \in I}$ and $\{Y_i\}_{i \in I}$ are *computationally indistinguishable*, denoted by $\{X_i\}_{i \in I} \equiv_{\mathrm{C}} \{Y_i\}_{i \in I}$, if for every polynomial $p(\cdot)$ and every sufficiently large i, X_i and Y_i are indistinguishable by $p(|i|)$-sized circuits. An equivalent formulation is that $\{X_i\}_{i \in I}$ and $\{Y_i\}_{i \in I}$ are computationally indistinguishable if there exists a negligible function $\mu : \mathbb{N} \to [0,1]$ such that X_i and Y_i are indistinguishable by $\frac{1}{\mu(|i|)}$-sized circuits. We will sometimes abuse notation and say that the two random variables X_i and Y_i are computationally indistinguishable, denoted by $X_i \equiv_{\mathrm{C}} Y_i$, when each of them is a part of a probability ensemble such that these ensembles $\{X_i\}_{i \in I}$ and $\{Y_i\}_{i \in I}$ are computationally indistinguishable. We will also sometimes drop the index i from a random variable if it can be inferred from the context. In most of this cases, the index i will be of the form 1^n where n is called the *security parameter*.

2.2 Cryptographic assumptions.

For the purposes of this presentation, we will assume the existence of collision-resistant hash functions that are secure against circuits of sub-exponential size (i.e. 2^{n^ϵ} for some fixed $\epsilon > 0$).[10] Using an appropriate setting of the security parameter we will assume that all cryptographic primitives we use are secure against 2^{n^5}-sized circuits, where n is the security parameter for our protocol.[11] In contrast we aim to prove that our protocol is secure only against adversaries that use *uniform* probabilistic polynomial-time algorithms.[12]

[10] As mentioned in Section 1.5, our protocol can be proven secure under somewhat weaker assumptions at the cost of a more complicated analysis, see Section **[sec:complexity]**.

[11] For example, if we have a primitive that is secure against 2^{m^ϵ} sized circuit with security parameter m, then when given the security parameter n as input, we will invoke the primitive with $m = n^{5/\epsilon}$.

[12] The main limitation of the current protocol is that it is only secure against *uniform* adversaries rather than the quantitative difference (2^{n^5} vs. polynomial-time) between the hardness assumption and the adversary's running time. Indeed, our protocol is in fact secure against uniform 2^{n^δ}-time algorithms for some $\delta > 0$. However, for the sake of clarity, we chose to model the adversary as a uniform probabilistic *polynomial-time* algorithm. The protocol of Section 4 is also in fact secure against 2^{n^δ}-sized circuits for some $\delta > 0$.

2.3 Cryptographic Primitives

2.3.1 Commitment Schemes

We let Com denote a computational (i.e., statistically binding) commitment scheme (see [Gol01, Sec. 4.4.1] for definitions). That is, we denote a commitment to x by $\mathsf{Com}(x) = \mathsf{Com}(x; U_m)$. Note that we assume for simplicity that the commitment scheme Com is non-interactive, as the scheme of [Blu82] or [BOV03]. We can also use the 2-round scheme of [Nao89], that can be based on any one-way function. If Com is a statistically binding commitment scheme and α is the transcript of the execution of Com, then we denote by $\mathsf{com\text{-}value}(\alpha)$ the unique value committed to in the transcript α, and we let $\mathsf{com\text{-}value}(\alpha) = \bot$ if no such value exists.[13] Note that the function $\mathsf{com\text{-}value}(\cdot)$ can not be efficiently computed, but nonetheless is well defined. We say that a commitment scheme has *public decommitment* if in the decommitment (or reveal) phase, the receiver does not need to access its random tape from the commitment phase. In other words, the transcript from the commit phase and the sender's decommitment message are enough to efficiently obtain the commitment value $\mathsf{com\text{-}value}(\cdot)$ (and thus the sender can prove assertions about the committed value without access to the receiver's random tape). In this paper, we only consider commitment schemes that satisfy the public decommitment property.

Commit-with-extract schemes. Loosely speaking, a *commit-with-extract* scheme [BL02] is a commitment scheme where the sender also proves the knowledge of the committed value. The formal definition of commit-with-extract follows the definition of "witness-extended emulation" [Lin01] and requires that there exists an extractor that can simulate the view of the sender, and output a committed value that is compatible with this view. The resulting definition is the following:

Definition 2.1 (commit with extract)**.** A statistically binding commitment scheme $\mathsf{Comm\text{-}Ext}$ with sender A and receiver B is a *commit-with-extract* scheme if the following holds: there exists a probabilistic polynomial-time commitment extractor CK such that for every probabilistic polynomial-time committing party A^* and for every $x, y, r \in \{0,1\}^*$, upon input $(\mathsf{desc}(A^*), 1^t, x, y, r)$, where t is a bound on the running time of $A^*(x, y, r)$, machine CK outputs a pair, denoted $(CK_1, CK_2) = (CK_1(\mathsf{desc}(A^*), 1^t, x, y, r), CK_2(\mathsf{desc}(A^*), 1^t, x, y, r))$, satisfying the following conditions:

1. $\left\{CK_1(\mathsf{desc}(A^*), 1^t, x, y, r)\right\}_{x,y,r\in\{0,1\}^*} \equiv_{\mathrm{C}} \left\{\mathsf{view}_{A^*}(A^*(x, y, r), B)\right\}_{x,y,r\in\{0,1\}^*}$
2. $\Pr[CK_2(\mathsf{desc}(A^*), 1^t, x, y, r) = \mathsf{com\text{-}value}(CK_1) \text{ or } \mathsf{com\text{-}value}(CK_1) = \bot] > 1 - \mu(|x|)$

A commit-with-extract scheme is called *liberal* if the extractor CK runs in *expected* probabilistic-polynomial-time.

A constant-round liberal commit-with-extract scheme can be obtained based on any one-way-function by sequentially executing first a standard commitment scheme and then a constant-round zero-knowledge proof-of-knowledge for **NP** (e.g., the system of [FS89]) to prove knowledge of the commitment value. The commitment extractor CK will use the knowledge extractor of the commitment scheme. In this way, one obtains a *black-box* commitment extractor that rewinds the

[13] In case Com is only statistically binding, and not perfectly binding, then we let $\mathsf{com\text{-}value}(\alpha) = \bot$ also in the case that α does not determine a unique committed value. Note that this case can only occur with negligible probability.

sender A in order to obtain the committed value. In [BL02], a different construction of a commit-with-extract is given, that has a *strict* probabilistic polynomial-time *non-black-box* extractor. For simplicity of exposition we will assume throughout the paper that our commit-with-extract scheme has a *black-box* extractor that utilizes rewinding. However, one can use also a non-black-box scheme such as the one of [BL02] in our construction.

Notation for commitment schemes. We shall sometimes say that a party *sends* a commitment (or commit-with-extract) to a value x, and use the notation $\mathsf{Com}(x)$ or $\mathsf{Comm\text{-}Ext}(x)$, even though in actuality the scheme Com or $\mathsf{Comm\text{-}Ext}$ may be an *interactive* protocol.

2.3.2 Zero-knowledge

We will use the notion of *zero-knowledge proofs* and *argument* systems [GMR85] and the notion of *proofs of knowledge* [FFS87, TW87, BG93] (see [Chap. 4][Gol01] for definitions). In particular we will use the fact that if one-way functions exist then every language in **NP** has a constant-round zero-knowledge argument that is also a proof of knowledge [FS89].

2.3.3 Universal arguments

Universal arguments were defined in [BG02]. They are a variant of (interactive) CS proofs [Mic94, Kil92]. Loosely speaking, a *universal argument* is an interactive argument of knowledge for proving membership in **NEXP** (instead of **NP**).[14] A formal definition of universal arguments can be found in Appendix A. We will use a universal argument system that is also zero-knowledge ($ZKUARG$). There are known constructions of such systems that use a constant number of rounds rounds under the assumption that collision-resistent hash functions exist [BG02].

3 A Uniform Non-Malleable Coin-Tossing Protocol

In this section we will construct a non-malleable coin-tossing protocol. The protocol of this section has two limitation: The first limitation is that our protocol will only secure against adversaries that use *uniform* probabilistic polynomial-time algorithms (rather than polynomial-sized circuits or equivalently, polynomial-time algorithms with auxiliary input). The second limitation is that our protocol will only be secure against adversaries that use the *synchronizing* scheduling. As mentioned in Section 1.5, constructing a non-malleable coin-tossing protocol against synchronizing adversaries is sufficient for our desired applications (since in Section 6 we show how to transform a non-malleable zero-knowledge or commitment scheme secure against synchronizing adversaries into a scheme secure against general adversaries.)

Therefore it is the first limitation (security only against uniform adversaries) that is actually more serious. Note that usually in cryptography, it *is* possible to derive security against non-uniform adversaries from a security proof against uniform adversaries. This is because most proofs of security use only *black-box* reductions. However, this is *not* the case here, since we will use some diagonalization arguments in our proof of security that do not carry over to the non-uniform case.

[14]We use universal arguments rather than an interactive CS proof system because: (a) We need to use the proof-of-knowledge property of universal arguments. (b) CS proofs seem to inherently require a super-polynomial hardness assumption; Although we do use such an assumption in the current presentation, it is not clear whether or not such an assumption is inherent in our approach.

Nonetheless, in Section 4 we do construct a different non-malleable coin-tossing protocol that is secure against *non-uniform* adversaries.

Rough outline of proof structure. The general form of our non-malleable coin-tossing protocol is similar to previous (malleable) coin-tossing protocols. In fact, it is quite similar to a coin-tossing protocol of Lindell [Lin01], except for the following modification: The modification is that while the protocol of [Lin01] involves a zero-knowledge proof that some condition X occurs, in our protocol we prove that *either* X or Y occurs, where Y is some "bogus" condition that almost always will *not* be satisfied in a real execution. This is a technique that originated in the work of Feige, Lapidot and Shamir [FLS99], and has been used in several places since (e.g., [RK99, Bar01]). When this technique is used it is usually the case that one can ensure that Condition Y occurs if one has the power to "rewind" the adversary. Thus, it is usually the case that the *adversary simulator* ensures that Condition Y occurs in the simulation.[15] This will *not* be the case here. Although we do need to provide an adversary simulator (or equivalently, an ideal adversary) $\widehat{C}$ to satisfy Definition 1.3, our simulator will *not* use the bogus Condition Y and in fact Condition Y will not occur even in the simulation. In fact, Condition Y will be of a form that no polynomial-time algorithm will be able to ensure it occurs, even with the use of rewinding. If we're not using this condition, then what do we need it for? The answer is that we will use this condition in the security proof. In order to show that our actual simulator $\widehat{C}$ does satisfy the conditions of Definition 1.3 we will construct an "imaginary simulator" $\widehat{C}''$. This "imaginary simulator" will run in time that is super-polynomial and will use this long running time instead of rewinding to ensure that Condition Y occurs.[16] Using the output of this "imaginary simulator" as an intermediate hybrid we will be able to prove that our actual simulator satisfies the conditions of Definition 1.3.

3.1 Evasive Sets

We will need to use the existence of an (exponential-time constructible) set $R \subseteq \{0,1\}^*$ that is both pseudorandom and hard to hit in the following sense:[17]

Definition 3.1 (Evasive set)**.** Let $R \subseteq \{0,1\}^*$. For any $n \in \mathbb{N}$ denote $R_n \overset{def}{=} R \cap \{0,1\}^n$. We say that R is *evasive* if the following conditions hold with respect to some negligible function $\mu(\cdot)$:

Constructibility: For any $n \in \mathbb{N}$, the set R_n can be constructed in time 2^{n^3}. That is, there exists a 2^{n^3} time Turing machine M_R that on input 1^n outputs a list of all the elements in the set R_n. In particular this means that deciding whether or not $r \in R$ can be done in time $2^{|r|^3}$.

Pseudorandomness: The set R is pseudorandom against uniform probabilistic polynomial-time algorithms. That is, for all probabilistic polynomial-time Turing machines M, it holds that

$$\left| \Pr_{r \leftarrow_{\mathrm{R}} R_n}[M(r){=}1] - \Pr_{r \leftarrow_{\mathrm{R}} \{0,1\}^n}[M(r){=}1] \right| < \mathsf{neg}(n)$$

.

[15] This is the case also in [Bar01], although there the simulator used the knowledge of the adversary's code instead of rewinding to ensure that Condition Y occurs.

[16] In the non-uniform version of our protocol, the "imaginary simulator" will need to use also the knowledge of the adversary's code (in addition to a longer running time) to ensure that this condition occurs. See Section 4.

[17] For simplicity we "hardwired" into Definition 3.1 the constants and time-bounds required for our application.

Evasiveness: It is hard for probabilistic polynomial-time algorithms to find an element in R_n. Furthermore, even when given an element $r \in R_n$, it is hard for such algorithms to find a (different) element in R_n. Formally, for any probabilistic polynomial-time Turing machine M and for any $r \in R_n$,

$$\Pr[M(r) \in R_n \setminus \{r\}] < \mathsf{neg}(n)$$

Sets with very similar properties have been shown to exist by Goldreich and Krawczyk [GK92]. Using similar methods we can prove

Theorem 3.2. *Suppose that 2^{n^ϵ}-strong one-way-functions exist, then there exists an evasive set.*

Proof. First note that when constructing the set R_n it is enough to ensure that R_n satisfies the pseudorandomness and evasiveness properties only for probabilistic Turing machines whose description is of size at most $\log n$ and whose running-time[18] is at most $n^{\log n}$. This will ensure that the set $R = \cup_{n\in\mathbb{N}} R_n$ will be pesudorandom and evasive for all probabilistic polynomial-time Turing machines. We denote the set of Turing machines that have size at most $\log n$ and running time halted at $n^{\log n}$ by $\mathcal{M}_n$. Note that $|\mathcal{M}_n| \leq n$.

We fix $f : \mathbb{N} \to \mathbb{N}$ be a (polynomial-time computable) function such that $f(\cdot)$ is super-polynomial and $f(n) = 2^{o(n^\epsilon)}$. For concreteness, we will set $f(n) = n^{\log n}$. Suppose that we choose at random a subset $S = \{x_1, \dots, x_f\}$ of size $f(n)$ (That is, $x_1, \dots, x_f$ are chosen uniformly and independently in $\{0,1\}^n$). Let $\mu(n) \overset{def}{=} f(n)^{-1/3}$. Note that $\mu(\cdot)$ is a negligible function. By the chernoff inequality, for every Turing machine M, the probability (over the choice of S) that $|\,\mathsf{E}_{x\leftarrow_{\mathrm{R}}\{0,1\}^n}[M(x)] - \mathsf{E}_{x\leftarrow_{\mathrm{R}}S}[M(x)]| > \mu(n)$ is extremely low ($2^{-\Omega(f(n)^{1/3})}$). Thus with high probability S is pseudorandom for all the machines $M \in \mathcal{M}_n$.

Let $i \neq j \in [f]$ and let $M \in \mathcal{M}_n$. The probability over S that $\Pr[M(x_i) = x_j] > \mu(n)$ is at most $\frac{2^{-n}}{\mu(n)}$ (where $\mu(\cdot)$, as before is defined by $\mu(n) \overset{def}{=} f(n)^{-1/3}$). By taking a union bound over all possible such pairs (i,j) we see that with the overwhelming probability of at least $1 - \frac{2^{-n}f(n)^2}{\mu(n)}$, the set S will be evasive for all the machines $M \in \mathcal{M}_n$.

Under our assumptions, there exists a generator $G : \{0,1\}^{\mathrm{polylog}(n)} \to \{0,1\}^{n^{\log n}}$ such that $G(U_n)$ is pseudorandom for circuits of size at most $n^{\log^2 n}$. That is, for any circuit with input $m = m(n)$ and size at most $2^{n^{\epsilon'}}$,

$$\| \Pr[C(U_m) = 1] - \Pr[C(G(U_n)) = 1] \| < n^{-\log^2 n}$$

.

Given a set $S \subseteq \{0,1\}^n$ of size $f(n)$ we can use this generator to verify in deterministic $2^{\mathrm{polylog}(n)}$-time that it is indeed satisfies the pseudorandomness and evasiveness properties for the machines in $\mathcal{M}_n$. Indeed, this can be done by using the generator to decrease the number of coin tosses of all Turing machines in $\mathcal{M}_n$ to $\mathrm{polylog}(n)$ and then simulating all machines in $\mathcal{M}_n$ in time $2^{\mathrm{polylog}(n)}$ (since enumerating all possible $\mathrm{polylog}(n)$ coin tosses for each machine can be done in time $2^{\mathrm{polylog}(n)}$). This means that we have a deterministic $2^{\log^c(n)}$-time test T (for some constant $c > 0$) such that given a set $S = \{x_1, \dots, x_f\}$, if T outputs 1 then S is evasive, and $\Pr[T(U_{n\cdot f(n)} = 1] > 1 - \mu'(n)$ for some negligible function $\mu'(\cdot)$.

[18]We can assume that all Turing machines are "clocked". That is, they are of the form that first computes 1^t and then runs for at most t steps.

Under our assumptions, there exists also a pseudorandom generator $G' : \{0,1\}^{\mathrm{polylog}(n)} \rightarrow \{0,1\}^{2^{\log^c(n)}}$. We use this generator G' to *find* a set S that satisfies these properties in time $2^{\mathrm{polylog}(n)}$ (which is at most $2^{O(n)}$) in the following way: go over all possible seeds and stop at the lexicographically first seed s such that $T(G(s)) = 1$.

Note that for our purposes it is enough to use a pseudorandom generator that runs in time that is *exponential* in its seed length. Such generators exist under weaker conditions than the ones stated in the theorem [NW88, IW97]. $\square$

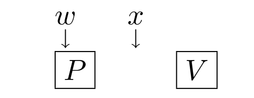
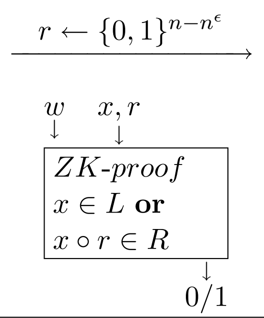
3.2 A Simple Simulation-Sound Proof System

In this section, we sketch the construction and proof of a simple simulation sound proof system that is based on the notion of evasive sets. Loosely speaking, a proof system is *simulation sound* [Sah99] if even when simulating a man-in-the-middle adversary, we still are ensured (with very high probability) that if the adversary's proof in the simulated right session passes verification, then either statement is a copy of the statement proven in the left session, or the statement is true. This should hold even if the statement proven in the simulated left session is false. Note that if we use a standard (i.e., standalone) zero-knowledge proof system in the man-in-the-middle setting, it may be that when we simulate the left session, the adversary manages to prove a false statement in the right session. Simulation soundness is a weaker condition than non-malleability, but it is sufficient for some applications (e.g., [Sah99, DDO$^+$01, Lin03]). The protocol of this section is not used in any other place of this work. We include it here because it is a simple protocol that demonstrates our techniques. Its main drawback is that we do not know a simple generalization for it to the non-uniform model. Another small drawback is that it uses the subexponential hardness assumption in a much more essential way than our other constructions.

In this subsection (and only in this subsection) we will assume that there exists an evasive set R with the following additional property: there exists some $\epsilon > 0$ such that for every $x \in \{0,1\}^{n^\epsilon}$, the uniform distribution on $x \circ \{0,1\}^{n-n^\epsilon} \cap R_n$ is computationally indistinguishable from $x \circ U_{n-n^\epsilon}$, where R_n denotes $R \cap \{0,1\}^n$ and $\circ$ denotes the string concatenation operator. Note that under suitable assumptions, a variant of the construction of the previous section satisfies this additional property. (Since a random dense enough subset of $\{0,1\}^n$ will satisfy this property.)

Protocol 3.3 is our simulation-sound zero-knowledge proof system. We now sketch its analysis. Firstly, we note that, unlike all other proof systems considered in this work, the soundness condition of this system does *not* hold with respect to *non-uniform* polynomial-sized cheating provers. This might have an element $x \circ r \in R$ with $x \notin L$ "hardwired" into it. However this system is sound with respect to *uniform* polynomial-time provers, since such provers cannot sample an element from R.

Let C be a man-in-the-middle adversary for Protocol 3.3 that utilizes the synchronizing scheduling. To simulate C we will simply use the simulator for the zero-knowledge proof in the obvious way. That is, the simulator will simulate the first step by following the honest left strategy (i.e., send a random $r \leftarrow_{\mathrm{R}} \{0,1\}^{n-n^\epsilon}$) and then treat the adversary C and the right party as one combined verifier, and simulate the zero-knowledge proof with respect to this verifier. The output of this simulator will be indistinguishable from the view of C in a real interaction. However, because this simulator effectively "rewinds" the right party, it is not at all clear that soundness of the right session is preserved. To show that this is the case we construct an "auxiliary simulator". On input x, this "auxiliary simulator" will run in super-polynomial time to compute a random r such that $x \circ r \in R$. It will then use the *honest prover* algorithm to prove that either $x \in L$ or $x \circ r \in R$. The simulation soundness property holds for this "auxiliary simulator" because of the evasiveness

Public input: 1^n: security parameter, $x \in \{0,1\}^{n^\epsilon}$ (statement to be proved is "$x \in L$") **Prover's auxiliary input:** w (a witness that $x \in L$)	
Step P1 (Send r): Prover sends to verifier a random string r of length $n - n^\epsilon$ **Steps P,V2.x (ZK Proof):** Prover proves to verifier using its input w via a zero knowledge universal argument that either $x \in L$ or that $x \circ r \in R$ where R is the evasive set. Verifier accepts if proof is completed successfully.	

Protocol 3.3. A Simulation-Sound Zero-Knowledge Protocol

property of the set R: if the adversary does not copy exactly x and r to the right session, then since it cannot find another $\tilde{x}, \tilde{r}$ such that $\tilde{x} \circ \tilde{r}$ is in the set R, and since no rewinding of the right party is done during the simulation, it is forced to choose a statement $\tilde{x}$ such that $\tilde{x} \in L$. Now the output of the auxiliary simulator is indistinguishable from the output of the real simulator by 2^{n^δ}-time algorithm for some $\delta > 0$. If we scale the security parameter appropriately, we can ensure that deciding membership in L can be done in time less than 2^{n^δ} and therefore the simulation soundness condition must also hold for the real simulator.

Reflection. It is not hard to construct simulation-sound (or even non-malleable) zero-knowledge in a setting where all parties have access to some public trusted verification key of some signature scheme.[19] However, in our setting we obviously do not have access to any such key. In some sense one may view the construction of this section as bypassing this difficulty by using some form of a "keyless signature scheme". That is, one can view a string r such that $x \circ r \in R$ as a *signature* on the string x. Like a (one-time) signature, given a signature $x \circ r \in R$ it is hard to come up with a different signature $x' \circ r' \in R$. However, unlike standard signature schemes, the signing algorithm does not use knowledge of a private key (since no such key exists) but rather uses super-polynomial time.

3.3 The Actual Construction

Our non-malleable coin-tossing protocol is Protocol 3.4 (See Page 17).

As a first observation, note that R_n is a set that is hard to hit by probabilistic polynomial-time algorithms. Therefore for any such algorithm that plays the left side, with overwhelming probability, it will *not* be the case that $r \in R_n$. Thus when the right side is honest, the soundness of the zero-knowledge argument guarantees that with high probability $r = r_1 \oplus r_2$.[21] The reason that

[19]Indeed, see Protocol 5.4 for such a construction.
[20]This is similar to the coin-tossing protocol of [Lin01].
[21]Note that this will *not* be true if we allow the adversary to be a polynomial-sized circuit, that may have an

Public input: 1^n: security parameter	$\overset{1^n}{\downarrow}$ $\boxed{L}$ $\boxed{R}$
Steps L,R1.x (Commitment to r_1): Left party selects $r_1 \leftarrow_{\mathrm{R}} \{0,1\}^n$ and commits to it using a perfectly-binding commit-with-extract scheme. We denote the transcript of the commitment by τ_1. We let s_1 denote the randomness used by left party during this step.	$\boxed{\textsf{Comm-Ext}(r_1; s_1)} \Rightarrow$
Step R2 (Send r_2): The right party selects a string $r_2 \leftarrow_{\mathrm{R}} \{0,1\}^n$ and sends it.	$\xleftarrow{r_2 \leftarrow_{\mathrm{R}} \{0,1\}^n}$
Step L3 (Send r): The left party sends the value $r = r_1 \oplus r_2$. We stress that the left party does not reveal the decommitment of τ_1.[20])	$\xrightarrow{r = r_1 \oplus r_2}$
Steps L,R4.x (Prove that $r = r_1 \oplus r_2$): The left party proves, using a zero-knowledge universal-argument ($ZKUARG$), that either $r = r_1 \oplus r_2$ (where r_1 is the unique string committed to by the transcript τ_1) or $r \in R_n$.	$\boxed{\begin{array}{l} ZKUARG \\ r = r_1 \oplus r_2 \\ \textbf{or } r \in R_n \end{array}} \Rightarrow$
The result of the protocol is the string r. We will use the convention that if one of the parties aborts (or fails to provide a valid proof) then the other party determines the result of the protocol.	

The right column contains a schematic description of the protocol as defined in the left column.

Protocol 3.4. A non-malleable coin-tossing protocol for uniform adversaries we need to use a universal argument (rather than a standard zero-knowledge proof or argument system for **NP**) is that we are not guaranteed by Theorem 3.2 that $R \in \mathbf{NP}$, but rather only that $R \in \mathbf{Dtime}(2^{n^3})$.

It is not hard to verify that the strategies for both the left and right parties can be carried out by probabilistic polynomial-time algorithms. (Note that we use here the prover efficiency condition of universal arguments , inherited from CS proofs.)

In order to show that Protocol 3.4 is a secure non-malleable coin-tossing protocol as per Definition 1.3, one needs to show that for every adversary C (that uses the *synchronizing* scheduling) there exists an ideal adversary $\widehat{C}$ that simulates the execution of C in a MIM attack. Indeed, let C be a probabilistic polynomial-time algorithm, and consider the execution of Protocol 3.4 in the man-in-the-middle setting where C plays the part of the channel. This execution is depicted in Figure 4 (Page 4). Note that we use the tilde (i.e., $\tilde{}$) symbol to denote the messages of the right session. The messages sent by the right and left parties are computed according to the protocol while the messages sent by the channel are computed by C who may use any (efficiently computable) function of the previous messages.

We need to simulate C by an "ideal" adversary $\widehat{C}$. The ideal adversary $\widehat{C}$ gets as input $r^{(1)}, r^{(2)}$ and should have two outputs (b, τ). Recall that $b \in \{1, 2\}$ and τ is a string simulating the output of C. Without loss of generality we can assume that τ should simulate the *view* of C in the execution. Clearly this view includes the strings $r, \tilde{r}$ where r is the result of the coin-tossing protocol in the left session and $\tilde{r}$ is the result of the coin-tossing protocol in the right session. For the simulation to be successful, it must hold that $r = r^{(1)}$ and $\tilde{r}$ is either equal to $r^{(1)}$ or to $r^{(2)}$. If this holds then we can decide by examining τ whether b should equal 1 or 2 based on whether $\tilde{r} = r^{(1)}$ or $\tilde{r} = r^{(2)}$. We see that the output b is redundant and that to prove that Protocol 3.4 is a secure non-malleable coin-tossing protocol it is enough to prove the following theorem:

Theorem 3.5. *Suppose that C is a probabilistic polynomial-time algorithm describing the strategy for a synchronizing adversary for Protocol 3.4, then there exists an algorithm $\widehat{C}'$ (computable by a probabilistic expected polynomial-time Turing machine with oracle access to C) with a single output such that, if $r^{(1)}, r^{(2)}$ are chosen uniformly and independently in $\{0, 1\}^n$ then,*

1. *$\widehat{C}'(r^{(1)}, r^{(2)})$ is computationally indistinguishable from the view of C in a real execution of Protocol 3.4 in the man-in-the-middle setting.*

2. *Let $r, \tilde{r}$ be the result of the coin-tossing algorithm in the left and right sessions recorded in the transcript $\widehat{C}'(r^{(1)}, r^{(2)})$. Then, with overwhelming probability it is the case that $r = r^{(1)}$ and $\tilde{r} \in \{r^{(1)}, r^{(2)}\}$.*

3.4 Proof of Theorem 3.5

Let C be a probabilistic polynomial-time algorithms representing the strategy of a synchronizing adversary for Protocol 3.4. In order to prove Theorem 3.5 we need to construct an algorithm $\widehat{C}'$ that simulates C. Figure 5 (Page 19) contains a schematic description of Algorithm $\widehat{C}'$.

Operation of Algorithm $\widehat{C}'$. Algorithm $\widehat{C}'$ gets as input two strings $r^{(1)}, r^{(2)}$ that were chosen uniformly and independently at random and in addition it gets black-box access to algorithm C.

element of R_n hardwired into it.

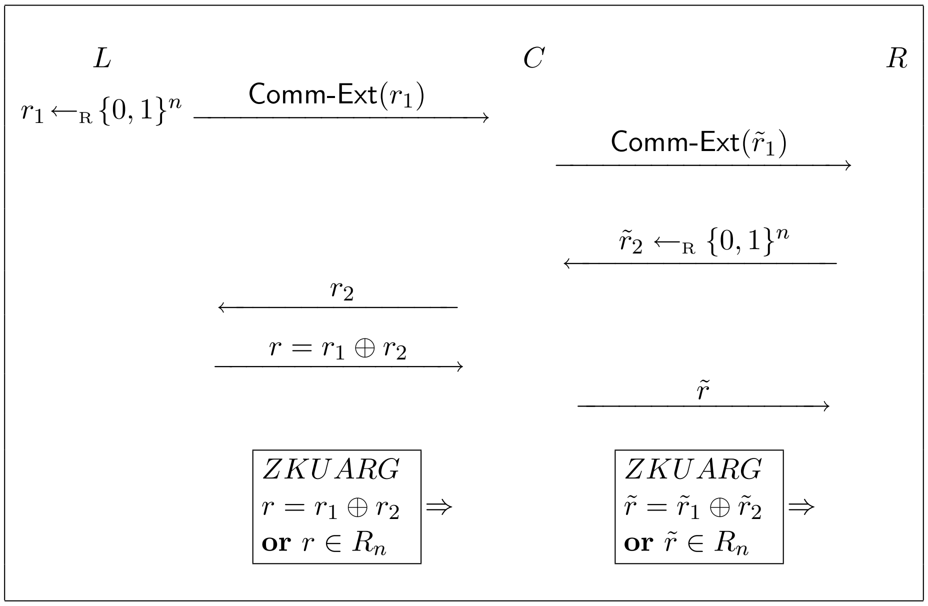
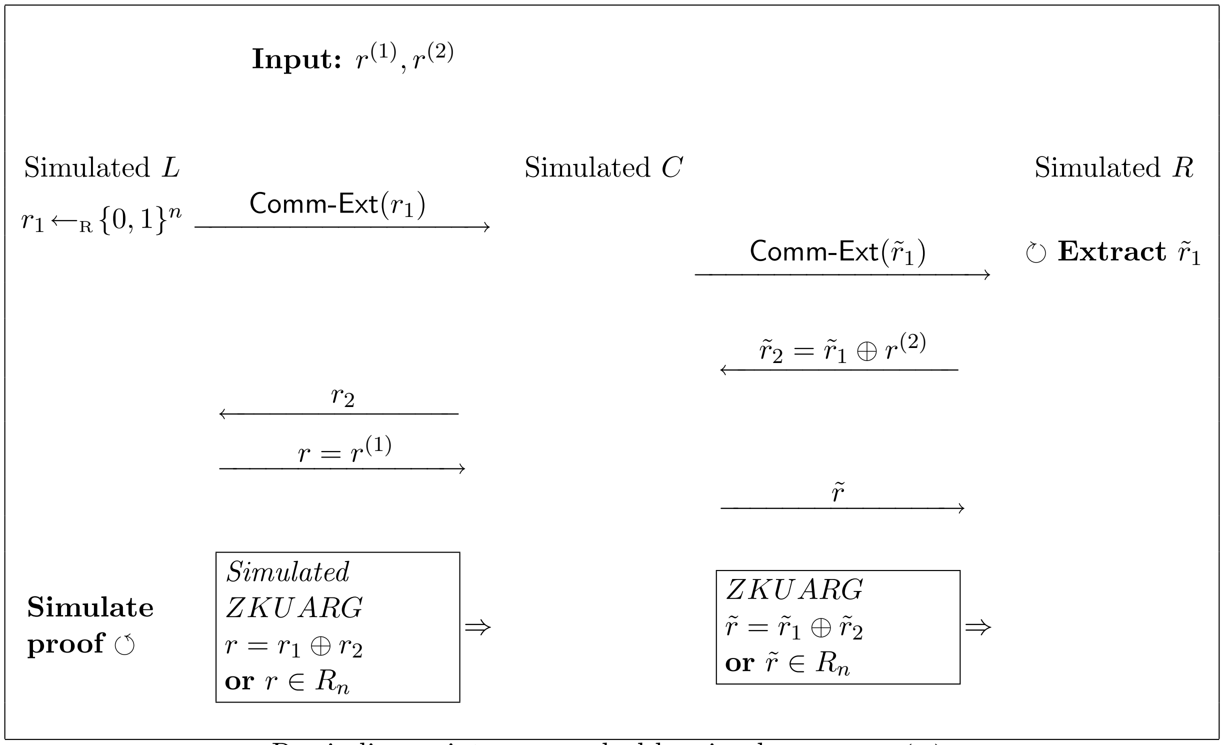

L
C
R
$r_1 \leftarrow_{\mathrm{R}} \{0,1\}^n$
$\mathsf{Comm\text{-}Ext}(r_1)$
$\mathsf{Comm\text{-}Ext}(\tilde{r}_1)$
$\tilde{r}_2 \leftarrow_{\mathrm{R}} \{0,1\}^n$
r_2
$r = r_1 \oplus r_2$
$\tilde{r}$
$ZKUARG$ $r = r_1 \oplus r_2$ **or** $r \in R_n$ $\Rightarrow$
$ZKUARG$ $\tilde{r} = \tilde{r}_1 \oplus \tilde{r}_2$ **or** $\tilde{r} \in R_n$ $\Rightarrow$

Figure 4: Execution of C in the man-in-the-middle setting


Input: $r^{(1)}, r^{(2)}$
Simulated L
Simulated C
Simulated R
$r_1 \leftarrow_{\mathrm{R}} \{0,1\}^n$
$\mathsf{Comm\text{-}Ext}(r_1)$
$\mathsf{Comm\text{-}Ext}(\tilde{r}_1)$
⟳ **Extract** $\tilde{r}_1$
$\tilde{r}_2 = \tilde{r}_1 \oplus r^{(2)}$
r_2
$r = r^{(1)}$
$\tilde{r}$
Simulate proof ⟲
Simulated $ZKUARG$ $r = r_1 \oplus r_2$ **or** $r \in R_n$ $\Rightarrow$
$ZKUARG$ $\tilde{r} = \tilde{r}_1 \oplus \tilde{r}_2$ **or** $\tilde{r} \in R_n$ $\Rightarrow$

Rewinding points are marked by circular arrows ⟲, ⟳

Figure 5: Algorithm $\widehat{C}'$ – simulation of C

Algorithm $\widehat{C}'$ emulates C's execution by running the honest left and right strategy, with some modifications as described below.

Algorithm $\widehat{C}'$ departs from the honest strategy in the right session (Steps L,R1.x) where it uses the *commitment extractor* of the commit-with-extract scheme Comm-Ext to simulate the execution of this phase, while also obtaining the value $\tilde{r_1}$ committed to by C. Note that in order to do that, $\widehat{C}'$ needs to treat both C and the left party L as a single combined sender for the commitment scheme and rewind them both at the same time.

The next modification is that instead of choosing $\tilde{r}_2$ to be a random string as is done by the honest right party, Algorithm $\widehat{C}'$ uses $\tilde{r}_2 = \tilde{r}_1 \oplus r^{(2)}$. Note however that this is still a uniform string because $r^{(2)}$ is uniformly distributed. In the left session in Step L3 $\widehat{C}'$ sends $r = r^{(1)}$ instead of $r = r_1 \oplus r_2$. This means that with high probability the statement $r = r_1 \oplus r_2$ is *false*. Also, with high probability it also holds that $r \notin R_n$. However, $\widehat{C}'$ uses the *simulator* for the $ZKUARG$ in order to simulate a proof for the false statement $r = r_1 \oplus r_2$ or $r \in R_n$. Similarly to the case when using the commitment extractor, Algorithm $\widehat{C}'$ will need to treat the adversary C and the honest right party as one combined verifier algorithm. Thus Algorithm $\widehat{C}'$ will also rewind the honest right party in this step. Note that all this can be carried out in expected polynomial-time.[22] Note that Algorithm $\widehat{C}'$ would not be well-defined if we needed to invoke the simulator and extractor at the same time, since it would mean that algorithm $\widehat{C}'$ would be rewinding itself. However, this can not happen since we assume the *synchronizing* scheduling.

Now that algorithm $\widehat{C}'$ is specified all that is left is to prove the two parts of Theorem 3.5. It turns out that Part 1 can be proven using the standard hybrid argument (relying on the security of the commitment scheme and zero-knowledge simulator). In contrast, the proof of Part 2 is much more complicated and it is for proving this part that we needed to introduce the evasive set R in the first place. We start with the proof of the more complicated part.

3.5 Proof of Theorem 3.5 Part 2

From a first impression, by looking at Figure 5 one may think that Part 2 of Theorem 3.5 should be easy to prove. After all, in the left session it is certainly the case that $r = r^{(1)}$ and the soundness of the universal-argument system used in the right session should ensure us that $\tilde{r} = \tilde{r}_1 \oplus \tilde{r}_2 = r^{(2)}$ (because it is unlikely that C can select $\tilde{r} \in R_n$). However, there is a caveat in this reasoning. The problem is that soundness is only ensured in an interactive setting where the prover only has access to a single interaction with the honest verifier. However, since Algorithm $\widehat{C}'$ is using the simulator in the left session, it will actually rewind also the verifier algorithm of the right party. he fact that Algorithm $\widehat{C}'$ gets the ability to rewind the verifier algorithm ruins our ability to argue about the soundness of the universal-argument system. Indeed, this problem is real (i.e., the soundness of the system may indeed be compromised). For example, consider an adversary that uses the *relaying* content strategy (i.e., copies all messages from the left session to the right session and vice versa). In an execution with such an adversary, it will be the case that $\tilde{r} = r$ and the proof in the right session will pass verification (since for such an adversary, the right session is identical to the left session). Because of the indistinguishability of our simulator, when we simulate the relaying adversary with Algorithm $\widehat{C}'$, also in the simulated transcript it holds that $\tilde{r} = r = r^{(1)}$. However, since $r^{(1)}$ is chosen at random, with overwhelming probability, it will *not* be the case

[22] Actually we can also have a *strict* polynomial-time non-black-box simulator by using the protocols of [Bar01] and [BL02].

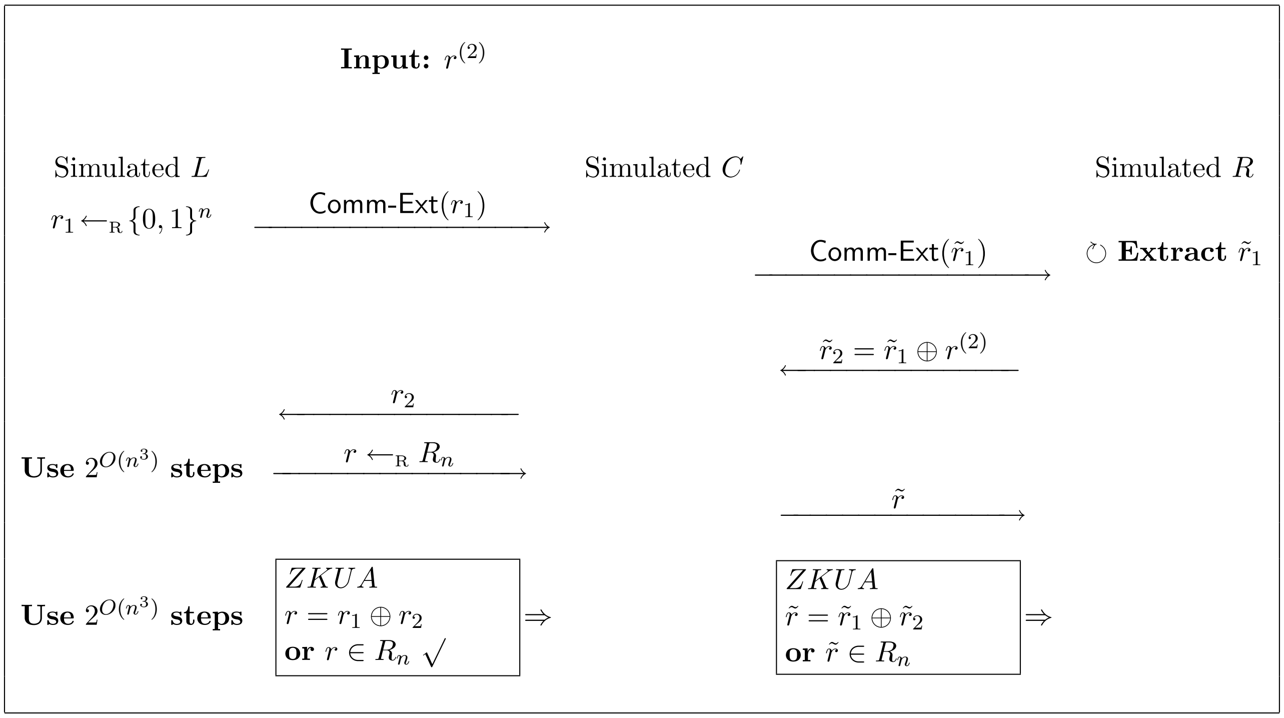

Input: $r^{(2)}$
Simulated L
Simulated C
Simulated R
$r_1 \leftarrow_R \{0,1\}^n$
$\mathsf{Comm\text{-}Ext}(r_1)$
$\mathsf{Comm\text{-}Ext}(\tilde{r}_1)$
Extract $\tilde{r}_1$
$\tilde{r}_2 = \tilde{r}_1 \oplus r^{(2)}$
r_2
Use $2^{O(n^3)}$ steps
$r \leftarrow_R R_n$
$\tilde{r}$
Use $2^{O(n^3)}$ steps
$ZKUA$ $r = r_1 \oplus r_2$ or $r \in R_n$ $\surd$
$ZKUA$ $\tilde{r} = \tilde{r}_1 \oplus \tilde{r}_2$ or $\tilde{r} \in R_n$

Figure 6: Algorithm $\widehat{C}''$

that $\tilde{r} = \tilde{r}_1 \oplus \tilde{r}_2$. Therefore, in the simulated transcript, the statement proven by the adversary in the right session will be *false*. Of course, this does not mean that Part 2 of Theorem 3.5 is false. Indeed, in the case of the relaying adversary it holds $\tilde{r} = r^{(1)}$ which is also allowed by the statement of Part 2. It just means that the naive approach to proving this part fails. We now turn to the actual proof.

We assume, for the sake of contradiction, that there exists a probabilistic polynomial-time algorithm C such that with non-negligible probability the corresponding ideal adversary $\widehat{C}'$ outputs a transcript where the result of right session $\tilde{r}$ is neither $r^{(1)}$ nor $r^{(2)}$. Note that in this case it must hold that the proof in the right session passes verification (or otherwise by our convention the right party can simply choose $\tilde{r} = r^{(2)}$). We consider the following $2^{O(n^3)}$-time algorithm $\widehat{C}''$ (depicted in Figure 6). Algorithm $\widehat{C}''$ behaves almost exactly as $\widehat{C}'$ with two differences:

1. In the left session (in Step L3) it chooses $r \leftarrow_R R_n$ instead of choosing $r = r^{(1)}$ (where $r^{(1)} \leftarrow_R \{0,1\}^n$). (This takes $2^{O(n^3)}$ steps using the constructibility property of R.)
2. Then, in the $ZKUA$ phase (Steps L,R4.x) Algorithm $\widehat{C}''$ does not use the zero-knowledge simulator but rather the *honest prover* algorithm of the universal argument to prove the true statement that either $r = r_1 \oplus r_2$ or $r \in R_n$. (This takes $2^{O(n^3)}$ steps.)

The output of $\widehat{C}''$ is computationally indistinguishable (by uniform algorithms) from the output of $\widehat{C}'$, even if the distinguisher is given $r^{(2)}$ (but not $r^{(1)}$). Indeed, this follows from the pseudorandomness of R_n and the zero-knowledge property of the universal-argument. Yet, combined with the hypothesis that in the output of $\widehat{C}'$ with non-negligible probability $\tilde{r} \notin \{r^{(1)} = r, r^{(2)}\}$, this implies that in the output of $\widehat{C}''$, with non-negligible probability it is the case that the following three conditions hold simultaneously:

1. $\tilde{r} \neq r$
2. $\tilde{r} \neq r^{(2)} = \tilde{r}_1 \oplus \tilde{r}_2$
3. The proof that either $\tilde{r} = \tilde{r}_1 \oplus \tilde{r}_2$ or $\tilde{r} \in R_n$ passes verification.

Now the soundness property of the universal arguments holds against 2^{n^5}-sized circuits. Since we are using less time than this and no rewinding/simulation is done during the relevant time[23] by $\widehat{C}''$ it can only be with negligible probability that the proof passes verification and the statement is false. This means that with non-negligible probability it will be the case that $\tilde{r} \neq r$ and $\tilde{r} \in R_n$ (since $\tilde{r} \neq \tilde{r}_1 \oplus \tilde{r}_2$). Now suppose that we halt the execution of $\widehat{C}''$ at the point where $\tilde{r}$ is computed. Up to this point $\widehat{C}''$ only needs to use a polynomial number of steps if it is given as input a random element $r \leftarrow_{\mathrm{R}} R_n$. This means that we have a probabilistic polynomial-time algorithm that gets a string $r \leftarrow_{\mathrm{R}} R_n$ and outputs a string $\tilde{r}$ that with non-negligible probability will be both different from r and a member of R_n. But this is clearly a contradiction to the evasiveness property of the set R_n. $\square$

3.6 Proof of Theorem 3.5 Part 1

We will now prove Part 1 of Theorem 3.5. That is, we prove the following claim:

Claim 3.6. *Let C be a probabilistic polynomial-time adversary strategy for Protocol 3.4. Let $\widehat{C}'$ be the simulator for C, as described in Section 3.4. Then, $\widehat{C}'(r^{(1)}, r^{(2)})$, where $r^{(1)}, r^{(2)} \leftarrow_R \{0,1\}^n$, is computationally indistinguishable from the view of C in a real execution of Protocol 3.4 in the man-in-the-middle setting.*

We prove the claim using the hybrid argument. We let $\mathbf{H}_0$ be the view of C in a real execution, and we let $\mathbf{H}_4$ be $\widehat{C}'(r^{(1)}, r^{(2)})$, where $r^{(1)}, r^{(2)} \leftarrow_{\mathrm{R}} \{0,1\}^n$. We prove that $\mathbf{H}_0 \equiv_{\mathrm{C}} \mathbf{H}_4$ by showing three intermediate random variables $\mathbf{H}_1, \mathbf{H}_2, \mathbf{H}_3$ such that

$$\mathbf{H}_0 \equiv_{\mathrm{C}} \mathbf{H}_1 \equiv_{\mathrm{C}} \mathbf{H}_2 \equiv_{\mathrm{C}} \mathbf{H}_3 \equiv_{\mathrm{C}} \mathbf{H}_4$$

We now describe these intermediate random variables, and show that indeed for every $1 \leq i \leq 4$, $\mathbf{H}_i$ is computationally indistinguishable from $\mathbf{H}_{i-1}$. To describe the random variable $\mathbf{H}_i$, we will describe the differences between it and the variable $\mathbf{H}_{i-1}$, and then show that the two variables are indistinguishable.

Hybrid $\mathbf{H}_1$: Simulated ZKUA. The difference between $\mathbf{H}_1$ and $\mathbf{H}_0$ is that in $\mathbf{H}_1$ we use the *simulator* of the zero-knowledge universal argument of Steps L,R4.x to simulate the view of the adversary C in these steps in the left session. Note that the verifier that is simulated is the combined strategy of C and the honest Right algorithm in these steps. The two random variables are indistinguishable by the zero-knowledge condition of the universal argument.

[23] $\widehat{C}''$ does use rewinding in the extraction stage but this is done before the *ZKUA* phase. Also in the extraction phase $\widehat{C}''$ only needs to rewind the honest left algorithm and not the honest right algorithm.

Hybrid $\mathbf{H}_2$: Choose $r = r^{(1)}$. The difference between $\mathbf{H}_2$ and $\mathbf{H}_1$ is that in $\mathbf{H}_2$ we choose the string r sent by the Left party in Step R2 of the left session to be $r^{(1)}$, where $r^{(1)}$ is chosen at random from $\{0,1\}^n$, instead of choosing r to be equal to $r_1 \oplus r_2$. The distributions $\mathbf{H}_2$ and $\mathbf{H}_1$ are computationally indistinguishable by the security of the commitment scheme used in Steps L,R1.x. (Note that in both hybrids, the coins used in the commitment scheme of these steps are not used anywhere else in the execution.)

Hybrid $\mathbf{H}_3$: Extract commitment. The difference between $\mathbf{H}_3$ and $\mathbf{H}_2$ is that in $\mathbf{H}_3$ we use the *commitment extractor* of the commit-with-extract scheme of Steps L,R1.x to simulate the view of the adversary C in these steps in the right session. Note that the sender that is simulated is the combined strategy of C and the honest Left algorithm in these steps. The two random variable are indistinguishable by the simulation condition of the commit-with-extract scheme (see Definition 2.1). Note that in addition to outputting a simulated view, the commitment extractor also outputs as an auxiliary outputs the value $\tilde{r}_1$ that is committed to by the adversary in this simulated view.

Hybrid $\mathbf{H}_4$: Send $\tilde{r}_2 = \tilde{r}_1 \oplus r^{(2)}$. The difference between $\mathbf{H}_4$ and $\mathbf{H}_3$ is that in $\mathbf{H}_4$, in Step R2 of the right session, we choose $\tilde{r}_2$ to be $\tilde{r}_1 \oplus r^{(2)}$, where $\tilde{r}_1$ is the value extracted by the commitment extractor, and $r^{(2)}$ is chosen at random. We note that even though we now choose $\tilde{r}_2$ in a different way, its distribution is still the uniform distribution (since $r^{(2)}$ is chosen at random). Therefore, the two variables $\mathbf{H}_4$ and $\mathbf{H}_3$ are identically distributed.

The proof is finished by observing that $\mathbf{H}_4$ is indeed the random variable $\widehat{C}'(r^{(1)}, r^{(2)})$, where $r^{(1)}, r^{(2)} \leftarrow_{\mathrm{R}} \{0,1\}^n$.

$\square$

4 Dealing with Non-Uniform Adversaries

In this section, we show how to modify Protocol 3.4 to obtain security even against adversaries that use *non-uniform* algorithms. Specifically, under the same assumptions we will construct a non-malleable coin-tossing protocol secure against *polynomial-sized* circuits (rather than probabilistic polynomial-time *uniform* algorithms ,as was done in Section 3). The protocol and its simulator will be quite similar to the uniform case. However, our proof of security will be somewhat different and will involve a *non-black-box* use of the adversary's code.

4.1 Evasive Set Families

An important component in our construction is a generalization of evasive sets as defined in Section 3.1, called *evasive set family*. Roughly speaking, an evasive set family is a family of sets indexed by strings, where each set is evasive (in the sense of Definition 3.1) with respect to algorithms that get its index as an advice string. The formal definition follows:[24]

[24] We have chosen to define evasive set families with respect to $n^{O(\log n)}$-time algorithms instead of polynomial-time. This change somewhat simplifies our exposition but is not really significant. In particular, we could have used any other fixed super-polynomial function.

Definition 4.1 (Evasive set family)**.** Let $\{R_{\alpha,n}\}_{\alpha\in\{0,1\}^*}$ be family of sets, where for any $\alpha \in \{0,1\}^*$, $R_\alpha \subseteq \{0,1\}^{n+p(n)}$ for some polynomial $p(\cdot)$. We say that the family $\{R_{\alpha,n}\}_{\alpha\in\{0,1\}^*}$ is an *evasive set family* if the following conditions hold with respect to some negligible function $\mu(\cdot)$:

Constructibility: There exists a Turing machine M such that $M(1^n, \alpha)$ runs for $|\alpha|2^{n^3}$ steps and outputs a list of all the elements in the set $R_{\alpha,n}$. In particular this means that deciding whether or not $r \in R_{\alpha,n}$ can be done in time $|\alpha|2^{|r|^3}$.

Pseudorandomness: The set $R_{\alpha,n}$ is pseudorandom against uniform probabilistic $n^{O(\log n)}$-time algorithms with advice α. That is, for any probabilistic polynomial-time Turing machine M it holds that

$$\left| \Pr_{r\circ s \leftarrow_{\mathrm{R}} R_{\alpha,n}}[M(\alpha, r\circ s) = 1] - \Pr_{r\circ s \leftarrow_{\mathrm{R}} \{0,1\}^{n+p(n)}}[M(\alpha, r\circ s) = 1] \right| < \mu(n)$$

Evasiveness: We say that $\alpha' \sim \alpha$ if there are two Turing machines M_1, M_2 of description $\log n$ and running time $n^{\log^2 n}$ such that $\Pr[M_1(\alpha) = \alpha'] > n^{-\log^2 n}$ and $\Pr[M_2(\alpha') = \alpha] > n^{-\log^2 n}$.

For every $\alpha' \sim \alpha$ and every probabilistic $n^{O(\log n)}$-time Turing machine M, if $r \circ s$ is chosen at random from $R_{\alpha,n}$, and denote $r' \circ s' = M(\alpha', r, s)$, the probability that $r' \neq r$ and $r' \circ s' \in R_{\alpha',n}$ is less than $\mu(n)$.

Erratum: In a previous version of this manuscript we assumed erroneously that $\sim$ is an equivalence relation.[25] This implied a simpler construction (and also slightly simpler definition) of evasive sets, since by moving from α to the lexicographically first member in the equivalence class of α, we could assume that for every $\alpha' \sim \alpha$ it holds that $R_{\alpha,n} = R_{\alpha',n}$. However, this is actually wrong, since the relation $\sim$ is symmetric, reflexive but not transitive. Therefore we need a different construction of evasive sets than the one suggested in the previous version.

As in the case of evasive sets, under the assumptions of this paper there exists an evasive set family. That is, we have the following theorem:

Theorem 4.2. *Suppose that 2^{n^ϵ}-strong one-way permutations exist, then there exists an evasive set family.*

(The one way permutations are only needed to get a non-interactive commitment scheme, alternatively we can use the one way functions based commitment scheme of [Nao89] at the cost of a slightly more complicated definition of evasive sets or the commitment scheme of [BOV03] at the cost of slightly stronger complexity assumptions.)

Proof. For every $\alpha \in \{0,1\}^*$ and $t \in \mathbb{N}$, define $e(\alpha, t) \in \{0,1\}^n$ to be a string such that:

- For every $100\log n$-length and t-time Turing machine M, $\Pr[M(\alpha) = e(\alpha, t)] < n^{-\log^2 n}$.
- $e(\alpha, t)$ can be computed in time $t \cdot n^{\mathrm{polylog}(n)}$.

[25]We note that it is possible to define an asymptotic version of $\sim$ that would be an equivalence relation, but this asymptotic version does not make sense for single finite strings (as opposed to sequences of strings). Confusion between these two versions seems to be the source of our previous error.

Choose k to be small enough power of n such that we can have two non-interactive commitments Com_w and Com_s with pseudorandom commitments such that Com_w is secure for $n^{\text{polylog}(n)}$ time and can be broken in time $2^{\sqrt{k}}$ and Com_s is secure for time $2^{k^{1.1}}$ and can be broken in time $2^{n^{1.5}}$. We let $G : \{0,1\}^k \to \{0,1\}^n$ be a pseudorandom generator strong against $n^{\text{polylog}(n)}$-size circuits that can also be broken in time $n^{\text{polylog}(n)}$. We also assume that G is one to one (we can obtain such a pseudorandom generator using an appropriate one-way permutation).

For every $u \in \{0,1\}^k$ we identify u with a number between 0 and $2^k - 1$. We define the set $R_{\alpha,n}$ as follows: $G(u) \circ c_1 \circ c_2$ where $c_1 = \mathsf{Com}_w(e(\alpha, u \cdot 2^{\sqrt{k}}), s_1)$ for some coins s_1 for the commitment scheme and $c_2 = \mathsf{Com}_s(e(\alpha, (2^k - u) \cdot 2^{n^{1.5}}), s_2)$ for some coins s_2. Constructibility follows by construction, and pseudorandomness follows by the security of the pseudorandom generator and commitment scheme. To show evasiveness, let $\alpha \sim \alpha'$ and suppose that if u, s_1, s_2 are chosen uniformly, then with noticeable probability

$$M(\alpha', \mathsf{Com}_w(e(\alpha, u \cdot 2^{\sqrt{k}}), s_1), \mathsf{Com}_s(e(\alpha, (2^k - u) \cdot 2^{n^{1.5}}), s_2)) = (r', c_1', c_2') \in R_{\alpha,n} \ , \qquad (1)$$

with $r' \neq G(u)$. In this case $r' = G(u')$ for some $u' \neq u$, and with noticeable probability either $u' > u$ or $u' < u$. In the first case, we obtain a contradiction by showing a machine N that on input α outputs the string $e(\alpha, u' \cdot 2^{\sqrt{k}})$ within less than $u' \cdot 2^{\sqrt{k}}$. The machine N will compute α' from α, run M on the inputs $(\alpha', G(u), \mathsf{Com}_w(e(\alpha, u \cdot 2^{\sqrt{k}}), U), \mathsf{Com}_s(0^n, U')$ (where U, U' denote random strings of appropriate lengths) and will output the string committed to using the weak commitment Com_w output by M (this can be done in $2^{\sqrt{k}}$ time). The key observation is that this process has to succeed with the same probability as it would have been if we didn't commit using Com_s to 0^n but rather to the right input as in Equation 1, because otherwise we get a contradiction to the security of Com_s. A similar reasoning applies in the case that $u' < u$ with noticeable probability, this time breaking Com_s instead. In this case we don't need to commit to 0^n in Com_w since we have enough time to compute $e(\alpha, u \cdot 2^{\sqrt{k}})$. □

4.2 The construction

Protocol 4.3 is our non-malleable coin-tossing protocol for non-uniform adversaries. It is very similar to the non-malleable coin-tossing protocol for uniform adversaries (Protocol 3.4). In fact, there are only two main changes:

First modification (adding a preliminary stage). We add a preliminary stage (Steps L,R 0.x) where each party sends a commitment to a hash of the all-zeros string and prove that it knows the hashed value using a zero-knowledge universal argument ($ZKUA$).[26]

Step L0.1.x - left commitment The left player commits to a hash of the all-zeros string and proves knowledge of the hashed value.

Step R0.1.1 (Right sends hash): Right party chooses a random collision-resistent hash function h_1 and sends it to the left party.

[26]We do not use a commit-with-extract scheme here because we need to prove knowledge of the hash's preimage and not knowledge of the value that is committed to.

Step L0.1.2 (Left commits): The left party a commitment to $h_1(0^n)$. That is, it chooses a string s_1 (random coins for the commitment) and sends $y_1 = \mathsf{Com}(h_1(0^n); s_1)$

Steps L,R0.1.2.x (Left proves knowledge): The left party proves using a $ZKUA$ that it knows a string α (where $|\alpha| \leq n^{\log n}$) and a string s_1 such that $y_1 = \mathsf{Com}(h_1(\alpha); s_1)$ where y_1 is the commitment sent at Step L0.1.1

Note: We stress that since we use a universal argument the length of the string α is *not* bounded by any fixed polynomial in the security parameter. However for convenience, we will require that $|\alpha| \leq n^{\log n}$. Note that if the left party follows the protocol then it would be the case that $\alpha = 0^n$.

Steps L,R0.2.x - right commitment These steps are almost a "mirror image" of Steps L,R0.1.x (with one slight difference). That is, the Right player now commits to a hash of the all-zeros string and proves knowledge of the hashed value.

Step R0.2.1 (Right commits): The right party chooses a random collision resistant hash function h_2 and sends h_2 along with a commitment to $h_2(0^n)$. That is, it chooses a string s_2 (random coins for the commitment) and sends h_2 and $y_2 = \mathsf{Com}(h_2(0^n); s_2)$.[27]

Steps L,R0.2.2.x (Right proves knowledge): The right party proves using a ZKUA that it knows a string β (where $|\beta| \leq n^{\log n}$) and a string s_2 such that $y_2 = \mathsf{Com}(h_2(\alpha); s_2)$ where y_2 is the commitment sent at Step R0.2.1

Second modification (modifying the $ZKUA$). In Step 5, the left party proves using the zero-knowledge universal-argument that either $r = r_1 \oplus r_2$ or $r \in R_{\alpha\circ\beta,n}$ where α is such that $y_1 = \mathsf{Com}(h_1(\alpha), s_1)$ for some s_1 and β is such that $y_2 = \mathsf{Com}(h_2(\beta), s_2)$ for some s_2. (Recall that $\alpha \circ \beta$ denotes the concatenation of α and β.)

By applying these changes we obtain Protocol 4.3 (see Page 27).

As in the uniform case (i.e., of Protocol 3.4), what we need to prove is the following Theorem (which is the non-uniform analog of Theorem 3.5):

Theorem 4.4. *Let $C_{\mathsf{n.u.}}$ be a polynomial-sized circuit (representing the adversary's strategy for Protocol 4.3). Then, there exists an algorithm $\widehat{C}'_{\mathsf{n.u.}}$ (that can be computed in expected probabilistic polynomial-time algorithm with oracle access to $C_{\mathsf{n.u.}}$) such that if $r^{(1)}, r^{(2)}$ are chosen uniformly and independently in $\{0,1\}^n$ then:*

[27]Note that this step is not an exact mirror image of the previous step since the Right party, that is the *sender* of the commitment, is choosing the hash function.

Public input: 1^n: security parameter	$\overset{1^n}{\downarrow}$ $\boxed{L}\qquad\boxed{R}$
Steps L,R0.1.x (Left commits to α): Right party chooses and sends a hash h_1. Left sends $y_1 = \mathsf{Com}(h_1(0^n))$ and then proves using a ZKUA that it knows a value α such that $y_1 = \mathsf{Com}(h_1(\alpha))$ (where $\lvert\alpha\rvert \le n^{\log n}$). **Steps L,R0.1.x (Right commits to β):** Right party chooses a hash h_2, and sends h_2 and $y_2 = \mathsf{Com}(h_2(0^n))$. It then proves using a ZKUA that it knows a value β such that $y_2 = \mathsf{Com}(h_2(\beta))$ (where $\lvert\beta\rvert \le n^{\log n}$).	$\xleftarrow{h_1}$ $\xrightarrow{h_1, y_1 = \mathsf{Com}(h_1(0^n))}$ $\boxed{\begin{array}{l} ZKUA \text{ of } \alpha, s \text{ s.t.} \\ y_1 = \mathsf{Com}(h_1(\alpha), s) \end{array}} \Rightarrow$ $\xleftarrow{h_2, y_2 = \mathsf{Com}(h_2(0^n))}$ $\Leftarrow \boxed{\begin{array}{l} ZKUA \text{ of } \beta, s \text{ s.t.} \\ y_2 = \mathsf{Com}(h_2(\beta), s) \end{array}}$
Continue as in Protocol 3.4 (The only change is in Steps L,R4.x)	
Steps L,R1.x (Commitment to r_1): *(unchanged)* Left party selects $r_1 \leftarrow_{\mathrm{R}} \{0,1\}^n$ and commits to it using a perfectly-binding commit-with-extract scheme. We denote the transcript of the commitment by τ_1. We let s_1 denote the randomness used by left party during this step. **Step R2 (Send r_2):** *(unchanged)* The right party selects a string $r_2 \leftarrow_{\mathrm{R}} \{0,1\}^n$ and sends it. **Step L3 (Send r):** *(unchanged)* The left party sends the value $r = r_1 \oplus r_2$ (without revealing the decommitment of τ_1). **Steps L,R4.x (Prove that $r = r_1 \oplus r_2$):** The left party proves using a zero-knowledge universal-argument ($ZKUA$) that either $r = r_1 \oplus r_2$ or $r \in R_{\alpha\circ\beta,n}$, where $y_1 = \mathsf{Com}(h_1(\alpha))$ and $y_2 = \mathsf{Com}(h_2(\beta))$.	$\boxed{\mathsf{Comm\text{-}Ext}(r_1; s_1)} \Rightarrow$ $\xleftarrow{r_2 \leftarrow_{\mathrm{R}} \{0,1\}^n}$ $\xrightarrow{r = r_1 \oplus r_2}$ $\boxed{\begin{array}{l} ZKUA \\ r = r_1 \oplus r_2 \\ \textbf{or } r \in R_{\alpha\circ\beta,n} \end{array}} \Rightarrow$
The result of the protocol is the string r. We will use the convention that if one of the parties aborts (or fails to provide a valid proof) then the other party determines the result of the protocol.	

Protocol 4.3. A non-malleable coin-tossing protocol for non-uniform adversaries

1. *$\widehat{C}'_{\mathsf{n.u.}}(r^{(1)}, r^{(2)})$ is computationally indistinguishable from the view of $C_{\mathsf{n.u.}}$ in a real execution of Protocol 4.3 with the honest left and right parties.*

2. *Let $r, \tilde{r}$ be the result of the coin-tossing protocol in the left and right sessions of the view $\widehat{C}'_{\mathsf{n.u.}}(r^{(1)}, r^{(2)})$. Then, with overwhelming probability it is the case that $r = r^{(1)}$ and $\tilde{r} \in \{r^{(1)}, r^{(2)}\}$.*

4.3 Proof of Theorem 4.4

To prove Theorem 4.4, consider a polynomial-sized circuit $C_{\mathsf{n.u.}}$ and consider the execution of Protocol 4.3 with $C_{\mathsf{n.u.}}$ playing the part of the channel. Such an execution is depicted in Figure 7.

We will use a simulator $\widehat{C}'_{\mathsf{n.u.}}$ that is very similar to the simulator used in the uniform case (see Section 3.4). In fact, the only change will be that we need to simulate also the preliminary steps (Steps L,R0.x). To simulate these steps, algorithm $\widehat{C}'_{\mathsf{n.u.}}$ will simply follow the honest left and right strategy (i.e., commit to a hash of the all zeros string). To simulate the other steps, we will follow the same strategy as the simulator $\widehat{C}'$, as described in Section 3.4 (see also Figure 5). That is, we will use the commitment extractor to extract $\tilde{r}_1$, choose $\tilde{r}_2$ to be $\tilde{r}_1 \oplus r^{(2)}$, and use the simulator of the zero-knowledge universal-argument to simulate Steps L,R4.x. Figure 8 (Page 30) contains a schematic description of the resulting algorithm $\widehat{C}'_{\mathsf{n.u.}}$.
As in the uniform case, to prove Theorem 4.4, we need to prove two Lemmas analogous to Parts 1 and 2 of Theorem 3.5:

Lemma 4.5. *For $r^{(1)}, r^{(2)} \leftarrow_R \{0,1\}^n$, $\widehat{C}'_{\mathsf{n.u.}}(r^{(1)}, r^{(2)})$ is computationally indistinguishable from the view of $C_{\mathsf{n.u.}}$ when executing Protocol 4.3 with the honest L and R in the man-in-the-middle-setting.*

Lemma 4.6. *For $r^{(1)}, r^{(2)} \leftarrow_R \{0,1\}^n$, with overwhelming probability $\widehat{C}'_{\mathsf{n.u.}}(r^{(1)}, r^{(2)})$ is a view of an execution of Protocol 4.3 where the result r of the left session is $r^{(1)}$ and the result $\tilde{r}$ of the right session is either $r^{(1)}$ or $r^{(2)}$.*

The proof of Lemma 4.5 is obtained by fairly standard hybrid arguments, and is almost identical to the proof of Theorem 3.5 Part 1.[28] Thus we omit this proof here. In contrast, the proof of Lemma 4.6 is more complicated and involves a *non-black-box* use of the code of the adversary $C_{\mathsf{n.u.}}$.

4.4 Proof of Lemma 4.6

The general outline proof of Lemma 4.6 follows the proof of Theorem 3.5 Part 2 (see Section 3.5). We assume, for the sake of contradiction, that there exists a polynomial-sized circuit $C_{\mathsf{n.u.}}$, such that with non-negligible probability in the transcript $\widehat{C}'_{\mathsf{n.u.}}(r^{(1)}, r^{(2)})$ it is the case that $\tilde{r} \notin \{r^{(1)}, r^{(2)}\}$ (where $\tilde{r}$ denotes the result string in the right session).[29] We now construct a $2^{O(n^3)}$-time algorithm $\widehat{C}''_{\mathsf{n.u.}}$ with one input that will have the following two properties:

1. For randomly chosen $r^{(2)} \leftarrow_{\mathrm{R}} \{0,1\}^n$, the random variable $(r^{(2)}, \widehat{C}''_{\mathsf{n.u.}}(r^{(2)}))$ is computationally indistinguishable by uniform algorithms from the random variable $(r^{(2)}, \widehat{C}'_{\mathsf{n.u.}}(r^{(1)}, r^{(2)}))$, where $r^{(1)}, r^{(2)} \leftarrow_{\mathrm{R}} \{0,1\}^n$.

[28]This is due to the fact that the only difference between the simulator $\widehat{C}'_{\mathsf{n.u.}}$ we present here and the simulator $\widehat{C}'$ of Section 3.4 is that Algorithm $\widehat{C}'_{\mathsf{n.u.}}$ needs to simulates also the preliminary phase of Steps L,R0.x. However, in these steps it uses the same strategy as used by the honest parties.

[29]Note that by the definition of Algorithm $\widehat{C}'_{\mathsf{n.u.}}$ it is always the case that $r = r^{(1)}$, where r denote the result string of the left session in $\widehat{C}'_{\mathsf{n.u.}}(r^{(1)}, r^{(2)})$.

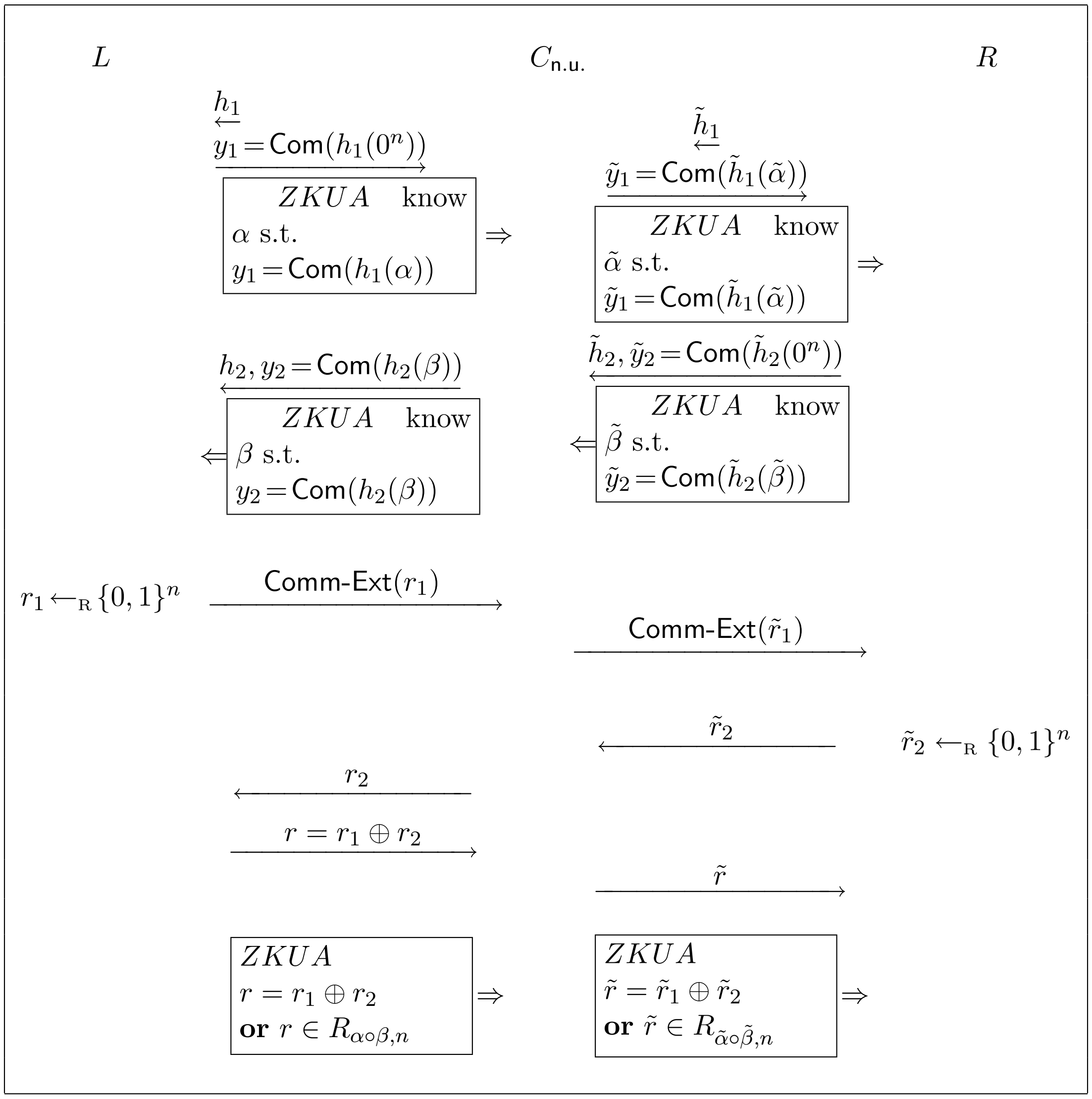

L
$C_{\mathsf{n.u.}}$
R
h_1
$y_1 = \mathsf{Com}(h_1(0^n))$
$ZKUA$ know α s.t. $y_1 = \mathsf{Com}(h_1(\alpha))$
$\tilde{h}_1$
$\tilde{y}_1 = \mathsf{Com}(\tilde{h}_1(\tilde{\alpha}))$
$ZKUA$ know $\tilde{\alpha}$ s.t. $\tilde{y}_1 = \mathsf{Com}(\tilde{h}_1(\tilde{\alpha}))$
$h_2, y_2 = \mathsf{Com}(h_2(\beta))$
$ZKUA$ know β s.t. $y_2 = \mathsf{Com}(h_2(\beta))$
$\tilde{h}_2, \tilde{y}_2 = \mathsf{Com}(\tilde{h}_2(0^n))$
$ZKUA$ know $\tilde{\beta}$ s.t. $\tilde{y}_2 = \mathsf{Com}(\tilde{h}_2(\tilde{\beta}))$
$r_1 \leftarrow_{\mathrm{R}} \{0,1\}^n$
$\mathsf{Comm\text{-}Ext}(r_1)$
$\mathsf{Comm\text{-}Ext}(\tilde{r}_1)$
$\tilde{r}_2$
$\tilde{r}_2 \leftarrow_{\mathrm{R}} \{0,1\}^n$
r_2
$r = r_1 \oplus r_2$
$\tilde{r}$
$ZKUA$ $r = r_1 \oplus r_2$ **or** $r \in R_{\alpha \circ \beta, n}$
$ZKUA$ $\tilde{r} = \tilde{r}_1 \oplus \tilde{r}_2$ **or** $\tilde{r} \in R_{\tilde{\alpha} \circ \tilde{\beta}, n}$

Figure 7: Execution of $C_{\mathsf{n.u.}}$ in the man-in-the-middle setting

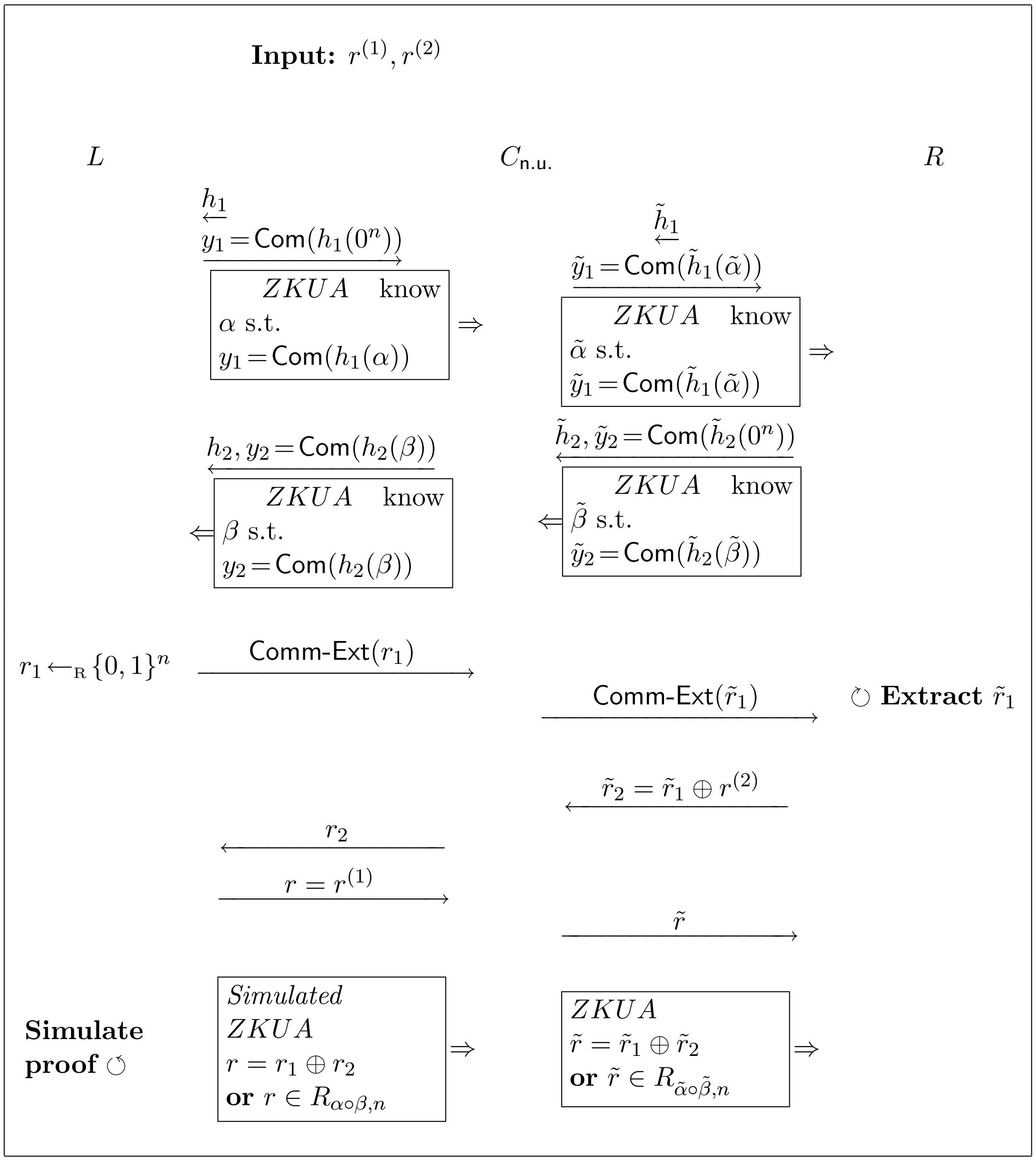

Input: $r^{(1)}, r^{(2)}$
L
$C_{\mathsf{n.u.}}$
R
h_1
$y_1 = \mathsf{Com}(h_1(0^n))$
$ZKUA$ know α s.t. $y_1 = \mathsf{Com}(h_1(\alpha))$
$\tilde{h}_1$
$\tilde{y}_1 = \mathsf{Com}(\tilde{h}_1(\tilde{\alpha}))$
$ZKUA$ know $\tilde{\alpha}$ s.t. $\tilde{y}_1 = \mathsf{Com}(\tilde{h}_1(\tilde{\alpha}))$
$h_2, y_2 = \mathsf{Com}(h_2(\beta))$
$ZKUA$ know β s.t. $y_2 = \mathsf{Com}(h_2(\beta))$
$\tilde{h}_2, \tilde{y}_2 = \mathsf{Com}(\tilde{h}_2(0^n))$
$ZKUA$ know $\tilde{\beta}$ s.t. $\tilde{y}_2 = \mathsf{Com}(\tilde{h}_2(\tilde{\beta}))$
$r_1 \leftarrow_{\mathrm{R}} \{0,1\}^n$
$\mathsf{Comm\text{-}Ext}(r_1)$
$\mathsf{Comm\text{-}Ext}(\tilde{r}_1)$
$\circlearrowleft$ **Extract** $\tilde{r}_1$
$\tilde{r}_2 = \tilde{r}_1 \oplus r^{(2)}$
r_2
$r = r^{(1)}$
$\tilde{r}$
Simulate proof $\circlearrowleft$
Simulated $ZKUA$ $r = r_1 \oplus r_2$ **or** $r \in R_{\alpha\circ\beta,n}$
$ZKUA$ $\tilde{r} = \tilde{r}_1 \oplus \tilde{r}_2$ **or** $\tilde{r} \in R_{\tilde{\alpha}\circ\tilde{\beta},n}$

Figure 8: Algorithm $\widehat{C}'_{\mathsf{n.u.}}$ - simulation of $C_{\mathsf{n.u.}}$

2. With overwhelming probability it is the case that in the output of $\widehat{C}''_{\mathsf{n.u.}}(r^{(2)})$ either $\tilde{r} = r$ or $\tilde{r} = r^{(2)}$, where r and $\tilde{r}$ denote the result of the coin-tossing protocol in the left and right session, respectively.

As in Section 3.5, the existence of an algorithm $\widehat{C}''_{\mathsf{n.u.}}$ satisfying these two properties directly leads to a contradiction. This is because they imply that also in the transcript $\widehat{C}'_{\mathsf{n.u.}}(r^{(1)}, r^{(2)})$ it is the case that with overwhelming probability $\tilde{r} \in \{r, r^{(2)}\}$ (note that this condition can be tested by a uniform algorithm). Since we know that in the transcript $\widehat{C}'_{\mathsf{n.u.}}(r^{(1)}, r^{(2)})$ it holds that $r = r^{(1)}$ we get that $\tilde{r} \in \{r^{(1)}, r^{(2)}\}$ and so the contradiction follows.

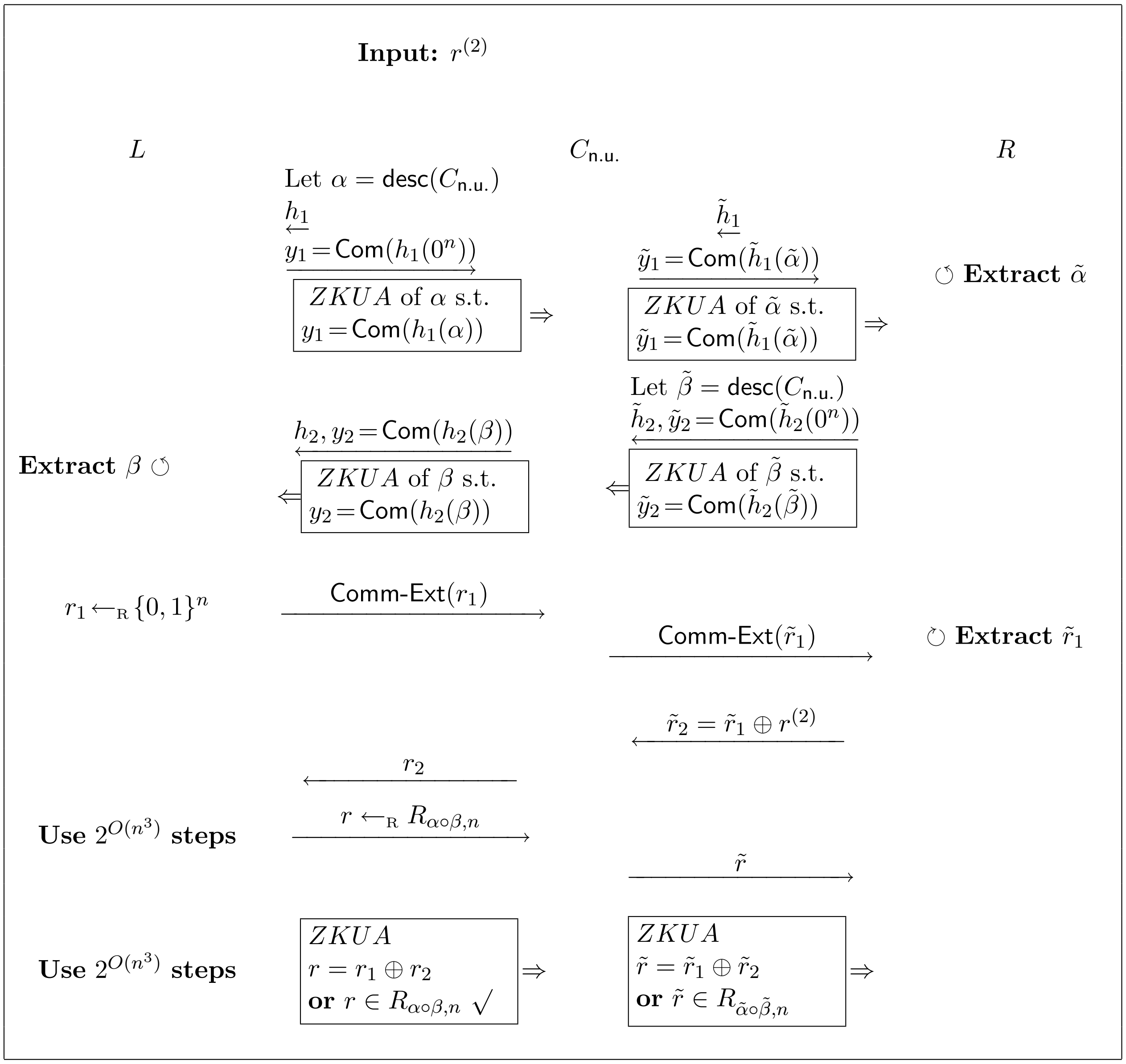

Input: $r^{(2)}$
L
$C_{\mathsf{n.u.}}$
R
Let $\alpha = \mathsf{desc}(C_{\mathsf{n.u.}})$
h_1
$y_1 = \mathsf{Com}(h_1(0^n))$
$ZKUA$ of α s.t. $y_1 = \mathsf{Com}(h_1(\alpha))$
$\tilde{h}_1$
$\tilde{y}_1 = \mathsf{Com}(\tilde{h}_1(\tilde{\alpha}))$
$ZKUA$ of $\tilde{\alpha}$ s.t. $\tilde{y}_1 = \mathsf{Com}(\tilde{h}_1(\tilde{\alpha}))$
Extract $\tilde{\alpha}$
Let $\tilde{\beta} = \mathsf{desc}(C_{\mathsf{n.u.}})$
$\tilde{h}_2, \tilde{y}_2 = \mathsf{Com}(\tilde{h}_2(0^n))$
$h_2, y_2 = \mathsf{Com}(h_2(\beta))$
Extract β
$ZKUA$ of β s.t. $y_2 = \mathsf{Com}(h_2(\beta))$
$ZKUA$ of $\tilde{\beta}$ s.t. $\tilde{y}_2 = \mathsf{Com}(\tilde{h}_2(\tilde{\beta}))$
$r_1 \leftarrow_{\mathrm{R}} \{0,1\}^n$
$\mathsf{Comm\text{-}Ext}(r_1)$
$\mathsf{Comm\text{-}Ext}(\tilde{r}_1)$
Extract $\tilde{r}_1$
$\tilde{r}_2 = \tilde{r}_1 \oplus r^{(2)}$
r_2
Use $2^{O(n^3)}$ **steps**
$r \leftarrow_{\mathrm{R}} R_{\alpha \circ \beta, n}$
$\tilde{r}$
Use $2^{O(n^3)}$ **steps**
$ZKUA$ $r = r_1 \oplus r_2$ **or** $r \in R_{\alpha \circ \beta, n}$ $\surd$
$ZKUA$ $\tilde{r} = \tilde{r}_1 \oplus \tilde{r}_2$ **or** $\tilde{r} \in R_{\tilde{\alpha} \circ \tilde{\beta}, n}$

Figure 9: Algorithm $\widehat{C}''_{\mathsf{n.u.}}$

What remains to be done is to describe the algorithm $\widehat{C}''_{\mathsf{n.u.}}$ and prove that it satisfies the two properties. Algorithm $\widehat{C}''_{\mathsf{n.u.}}$ is described in Figure 9 (Page 31); It follows the operation of Algorithm $\widehat{C}'_{\mathsf{n.u.}}$ with the following changes:

1. In Steps L,R0.1.x of the *left* session, Algorithm $\widehat{C}''_{\mathsf{n.u.}}$ sets the hashed string α to be the *description of* $C_{\mathsf{n.u.}}$*'s code* instead of $\alpha = 0^n$, which is what is done by the honest left party L and by Algorithm $\widehat{C}'_{\mathsf{n.u.}}$.

2. In Steps L,R0.2.x of the *right* session, Algorithm $\widehat{C}''_{\mathsf{n.u.}}$ follows the "mirror image" of the previous steps. That is, Algorithm $\widehat{C}''_{\mathsf{n.u.}}$ will set the hashed string $\tilde{\beta}$ to be the *description of* $C_{\mathsf{n.u.}}$*'s code* instead of $\tilde{\beta} = 0^n$ (as is done by the honest right party R and by Algorithm $\widehat{C}'_{\mathsf{n.u.}}$).

3. In the corresponding steps (Steps L,R0.2.x) of the *left* session, Algorithm $\widehat{C}''_{\mathsf{n.u.}}$ uses the *extractor* of the universal argument (this may take up to $n^{O(\log n)}$ steps) and extract a string β of length up to $n^{\log n}$ and a string s such that $\mathsf{Com}(\tilde{h_2}(\beta), s) = y_2$.

4. In Step L4 of the left session, algorithm $\widehat{C}''_{\mathsf{n.u.}}$ chooses r as a random element of the set $R_{\alpha\circ\beta,n}$, using $2^{O(n^3)}$ steps. (Recall that $\widehat{C}'_{\mathsf{n.u.}}$ used in this step $r = r^{(1)}$).

5. In Steps L,R5.x of the right session, algorithm $\widehat{C}''_{\mathsf{n.u.}}$ follows the honest prover algorithm for the $ZKUA$ system, and runs in $2^{O(n^3)}$ time to prove the true statement that $r \in R_{\alpha\circ\beta,n}$. (Recall that $\widehat{C}'$ used in this step the simulator for the $ZKUA$ system.)

Now that we described Algorithm $\widehat{C}''$, we need to show that it satisfies both Properties 1 and 2 mentioned above. We start with proving that it satisfies Property 2, since this is the more interesting part.

Property 2 We need to show is that with overwhelming probability it is the case that in the transcript $\widehat{C}''_{\mathsf{n.u.}}(r^{(2)})$, $\tilde{r} \in \{r, r^{(2)}\}$. Suppose that this is not the case. This means that with non-negligible probability $\tilde{r} \notin \{r, \tilde{r}_1 \oplus \tilde{r}_2\}$ (since it $\tilde{r}_1 \oplus \tilde{r}_2 = r^{(2)}$). Firstly, note that since we are using $2^{O(n^3)} = 2^{o(n^5)}$ time, and we are doing using no rewinding in Steps L,R5.x, the soundness of the ZKUA used in these steps in the right session, ensures us that with overwhelming probability either $\tilde{r} = \tilde{r}_1 \oplus \tilde{r}_2$ or $\tilde{r} \in R_{\tilde{\alpha}\circ\tilde{\beta},n}$. This means that with non-negligible probability it holds that $\tilde{r} \neq r$ but $\tilde{r} \in R_{\tilde{\alpha'}\circ\tilde{\beta}',n}$ for some $\tilde{\alpha'}, \tilde{\beta}'$ such that $y_1 = \mathsf{Com}(h_1(\tilde{\alpha'}))$, $y_2 = \mathsf{Com}(h_2(\tilde{\beta}'))$.

We note that with overwhelming probability $\tilde{\beta}' = \tilde{\beta} = \mathsf{desc}(C_{\mathsf{n.u.}})$. Indeed, otherwise (using the extractor for the universal argument) we would have a $2^{O(n^3)}$-time algorithm for breaking the hash function. We also note that the string $\tilde{\alpha'}$ can be computed with at least $n^{-\log n}$ probability from the string $\mathsf{desc}(C_{\mathsf{n.u.}})$ in $n^{O(logn)}$-time by applying the extractor to the ZKUA of Steps L,R0.1.3.x. (This procedure will indeed get the same string α', as otherwise, using also the knowledge extractor for the universal argument of Step L,R4.x, we would have a $2^{O(n^3)}$-time algorithm that finds collisions in the hash function with non-negligible probability.)

Because the string β extracted by Algorithm $\widehat{C}''_{\mathsf{n.u.}}$ is also computed in time $n^{O(\log n)}$ from $\mathsf{desc}(C_{\mathsf{n.u.}})$ and because $\alpha = \tilde{\beta} = \mathsf{desc}(C_{\mathsf{n.u.}})$, we get that the strings $\alpha \circ \beta$ and $\tilde{\alpha'} \circ \tilde{\beta}'$ are *equivalent*. Therefore, $R_{\alpha\circ\beta,n} = R_{\tilde{\alpha'}\circ\tilde{\beta}',n}$.

We see that if we halt the algorithm $\widehat{C}'_{\mathsf{n.u.}}$ after Step L3 we get a $n^{O(\log n)}$-time algorithm with advice $\alpha \circ \beta$ that given an element $r \in R_{\alpha\circ\beta,n}$ manages to output an element $\tilde{r} \in R_{\alpha\circ\beta,n} \setminus \{r\}$. By this we obtain a contradiction to the evasiveness property of this set family.

We note that since the set $R_{\alpha\circ\beta}$ is only evasive with respect to machines that get $\alpha\circ\beta$ as advice, it is crucial that Algorithm $\widehat{C}''_{\mathsf{n.u.}}$ used the description of the code of the adversary $C_{\mathsf{n.u.}}$ as α and $\tilde{\beta}$ in Steps L,R0.1,2.x.

Property 1. We now prove that the random variable $(r^{(2)}, \widehat{C}''_{\mathsf{n.u.}}(r^{(2)}))$ (where $r^{(2)} \leftarrow_{\mathrm{R}} \{0,1\}^n$) is computationally indistinguishable by uniform algorithms from the random variable $(r^{(2)}, \widehat{C}'_{\mathsf{n.u.}}(r^{(1)}, r^{(2)}))$ (where $r^{(1)}, r^{(2)} \leftarrow_{\mathrm{R}} \{0,1\}^n$). Intuitively, this follows from the secrecy of the commitment scheme, the pseudorandomness of the set $R_{\alpha\circ\beta,n}$, and the zero-knowledge property of the $ZKUA$. The actual proof, which utilizes the hybrid argument, follows.

We denote by $\mathbf{H}_0$ the random variable $(r^{(2)}, \widehat{C}'_{\mathsf{n.u.}}(r^{(1)}, r^{(2)}))$ and by $\mathbf{H}_4$ the random variable $(r^{(2)}, \widehat{C}''_{\mathsf{n.u.}}(r^{(2)}))$. We prove our claim by showing random variables $\mathbf{H}_1, \mathbf{H}_2, \mathbf{H}_3$ such that for every $1 \le i \le 4$, the variables $\mathbf{H}_i$ and $\mathbf{H}_{i-1}$ are computationally indistinguishable by probabilistic polynomial-time uniform algorithms.

We now describe these intermediate random variables, and show that indeed for every $1 \le i \le 4$, $\mathbf{H}_i$ is computationally indistinguishable from $\mathbf{H}_{i-1}$. To describe the random variable $\mathbf{H}_i$, we will describe the differences between it and the variable $\mathbf{H}_{i-1}$, and then show that the two variables are indistinguishable by a class that is at least as strong as probabilistic polynomial-time uniform algorithms.

Hybrid $\mathbf{H}_1$: Commit to $C_{\mathsf{n.u.}}$'s code. The difference between $\mathbf{H}_1$ and $\mathbf{H}_0$ is that in $\mathbf{H}_1$, the honest left and right parties commit to the code of $C_{\mathsf{n.u.}}$, instead of committing to the all zeros strings. That is, in our notation, $\alpha = \tilde{\beta} = \mathsf{desc}(C_{\mathsf{n.u.}})$. The variables $\mathbf{H}_1$ and $\mathbf{H}_0$ are computationally indistinguishable by polynomial-sized circuits by the security of the commitment scheme.

Hybrid $\mathbf{H}_2$: Extract β. The variable $\mathbf{H}_2$ is distributed identically to $\mathbf{H}_1$. However, in $\mathbf{H}_2$, after running the honest verifier in Steps L,R0.2.x of the left session, we use $n^{O(\log n)}$ time to run the knowledge extractor of the universal argument. We thus β that corresponds to the values hashed and committed to by the adversary the left session.

Hybrid $\mathbf{H}_3$: Choose $r \leftarrow_{\mathbf{R}} R_{\alpha\circ\beta,n}$. The difference between $\mathbf{H}_3$ and $\mathbf{H}_2$ is that in $\mathbf{H}_3$, the Left party chooses $r \leftarrow_{\mathrm{R}} R_{\alpha\circ\beta,n}$ instead of choosing $r = r^{(1)}$. The two hybrids are indistinguishable by probabilistic $n^{O(\log n)}$-time uniform algorithms by the pseudorandomness property of the evasive set family. Indeed, note that all the elements of these two random variable can be sampled using $n^{O(\log n)}$-time and using the description of $C_{\mathsf{n.u.}}$ as an advice string. This means that a $n^{O(\log n)}$-time uniform distinguisher between $\mathbf{H}_3$ and $\mathbf{H}_2$ can be converted into a $n^{O(\log n)}$-time with advice $C_{\mathsf{n.u.}}$ machine that distinguishes a random element of $R_{\alpha\circ\beta,n}$ from the uniform distribution on $\{0,1\}^n$. This is a contradiction to the pseudorandomness property of the evasive set family because $R_{\alpha\circ\beta,n} = R_{\mathsf{desc}(C_{\mathsf{n.u.}}),n}$. (This is due to the fact that the family is nice and β is computed $n^{O(\log n)}$-time computation from $\mathsf{desc}(C_{\mathsf{n.u.}})$, and hence $\alpha \circ \beta$ is *equivalent* to $\mathsf{desc}(C_{\mathsf{n.u.}})$.)

Hybrid $\mathbf{H}_4$: Use real and not simulated proof in Steps L,R4.x. The difference between $\mathbf{H}_4$ and $\mathbf{H}_3$ is that in $\mathbf{H}_4$, the left party follows the honest prover algorithm (that takes $2^{O(n^3)}$

time) to prove the true statement that either $r = r_1 \oplus r_2$ or $r \in R_{\alpha\circ\beta,n}$. The two variables are computationally indistinguishable by the zero-knowledge property of the universal argument.

The proof is finished by observing that $\mathbf{H}_4$ is in fact the random variable $(r^{(2)}, \widehat{C}''_{\mathsf{n.u.}}(r^{(2)}))$.

$\square$

5 Applications of Non-Malleable Coin-Tossing

In this section we construct a *constant-round* non-malleable commitment scheme and a *constant-round* non-malleable zero-knowledge argument system. This improves over the previously known schemes of [DDN91] that utilized a logarithmic number of rounds. The protocols of this section are only shown to be secure against man-in-the-middle adversaries that use the *synchronizing* scheduling. However, in Section 6 we show a generic way to transform such protocols into protocols that are secure also against *non-synchronizing* adversaries, thus finishing the proof of Theorem 1.5. We remark that our approach to proving Theorem 1.5 favors modularity and simplicity of presentation, at the expense of the resulting protocols' efficiency.

Our building blocks are our non-malleable coin-tossing protocol (which is secure in the plain model, without any setup assumption) and a non-malleable zero-knowledge argument in the *shared random string model*. This latter argument is a variant of the argument system of De Santis *et al.*[DDO$^+$01].[30] As noted in Section 1.5, the proof of Theorem 1.5 will go as follows: First we prove that the composition (using Construction 1.2) of a (plain model) non-malleable coin-tossing protocol with a (shared random string model) non-malleable zero-knowledge yields a non-malleable interactive zero-knowledge argument in the plain model. Then, we show how to use such an argument to obtain also a (plain model) non-malleable commitment scheme. In the course of the proof we will define and use a stronger form of non-malleability that we call *extractability in the MIM setting*. We believe that this notion (that appears already implicitly in [DDN91] and more explicitly in [DDO$^+$01]) is interesting in its own right.

5.1 Extractability in the MIM setting

Recall that the idea behind non-malleability is that an adversary in the MIM attack will not be able to utilize his interaction in one session to gain something in the other session. For example, consider the case of zero-knowledge proof systems. In such systems the left party plays the part of the *prover* and the right party plays the part of the *verifier*. We assume that in the left session the honest prover proves a statement x to the adversary, while in the right session the adversary proves a (possible related) statement $\tilde{x}$ to the honest verifier. We require that, unless $x = \tilde{x}$, if the adversary convinces the verifier then it could have done so even without interacting with the honest prover in the left session. In the non-interactive setting, [DDO$^+$01] makes a stronger requirement. They require that if the adversary can convince the verifier of the statement $\tilde{x}$ then the adversary can in fact output a *witness* for $\tilde{x}$. What this actually means is that in a non-malleable zero-knowledge system it is possible to simulate the left session using the zero-knowledge simulator and at the same time extract a witness in the right session using the knowledge extractor.

[30]We could also have used directly the system of [DDO$^+$01]. However, we choose to use a variant of that system that is secure under possibly weaker assumptions (does not assume the existence of dense cryptosystems). We remark that in contrast to the system of [DDO$^+$01], our zero-knowledge system is *interactive* (although still uses only a constant number of rounds).

This paradigm of "simulate from the left, extract from the right" also makes sense in other contexts. Consider *commitment schemes*, where the left party is the *sender* and the right party is the *receiver*. We would want to be able to simulate the left session, *without knowing the sender's input*,while at the same time extracting the committed value of the right session. In fact, although Dolev *et al.* [DDN91] do not make this extraction requirement part of their *definition* for non-malleable commitment scheme, they note that their *construction* for commitment-scheme has this property. As they note, this fact is important when they use their commitment scheme to construct a non-malleable zero-knowledge proof system.

We now provide a formal definition for *extractability in the MIM setting*. Intuitively, a protocol is extractable in the MIM setting if it can be "simulated from the left and extracted from the right". We assume that for any protocol a function (or relation) val is defined. This function takes a transcript and returns the "intended value" consistent with this transcript. For example, in the case of a (perfectly binding) commitment scheme, $\mathsf{val}(\sigma)$ is the unique value that is consistent with the transcript σ. In the case of a zero-knowledge system, val will be a relation rather than a function such that $y \in \mathsf{val}(\sigma)$ if y is a *witness* for the statement x proved in the transcript σ. In both cases we will let $\mathsf{val}(\sigma) = \perp$ if σ is an invalid or aborting transcript. We assume that given a transcript σ and a value y it is easy to determine whether $y \in \mathsf{val}(\sigma)$.[31]

We will assume that our protocols have a *determining message*. This is one message in the protocol that determines the value of the intended value. For example, in the case of a zero-knowledge system the determining message will be the statement x. In the case of a commitment scheme the determining message will be a message that determines the committed value uniquely. We now define a function (or relation) $\mathsf{i\text{-}value}$ that takes as input a transcript of *two concurrent sessions* τ. Let τ_L and τ_R denote the transcripts of the left and right sessions in τ. We define $\mathsf{i\text{-}value}(\tau) = \mathsf{val}(\tau_R)$ if the determining message in τ_R is *not* an exact copy of the determining message in τ_L. Otherwise, (if the determining messages are equal) we let $\mathsf{i\text{-}value}(\tau) = \perp$.

To simplify notations, we will assume that in our protocols only the left party gets an input (x, y). In commitment schemes this is always the case. In zero-knowledge systems we will assume that the left party gets (x, y) where y is a witness for x, and sends x as its first message. This allows to incorporate into the definition the ability of the adversary to choose the statement he will prove in the right session, based on information he gets in the left session. We assume that x is the *public* part of the input and y is the *secret* part of the input (e.g., in the case of commitment schemes x is empty). We will also assume that the adversary does not get an input, as this can be taken care of by non-uniformity.

We can now give a formal definition for extractability in the MIM setting:

Definition 5.1. Let $\Pi = (L, R)$ be a two party protocol with an intended value function $\mathsf{i\text{-}value}$ defined as above. We say that Π is *extractable in the MIM setting* if for any polynomial-sized MIM adversary C there exists a (standalone) simulator with 2-outputs $\widehat{C}$ such that for any inputs (x, y) to the honest left party:

1. If $(\tau', y') \leftarrow \widehat{C}(x)$ then with overwhelming probability either $\mathsf{i\text{-}value}(\tau') = \perp$ or $y' \in \mathsf{i\text{-}value}(\tau')$.

2. Let τ denote C's view when interacting with $L(x, y)$ and R, and let τ' denote the first output of $\widehat{C}(x)$. Then τ and τ' are computationally indistinguishable.

[31] This obviously holds in the case of zero-knowledge systems. In the case of commitment schemes we can ensure this holds by appending $\mathsf{val}(\sigma)$ with some auxiliary information

If $\mathsf{i\text{-}value}$ is a function (and not a relation) then we can make the following stronger requirement in addition to Items 1 and 2: We say that an extractable scheme Π is *strongly extractable* (in the MIM setting) if the distribution $(\tau, \mathsf{i\text{-}value}(\tau))$ conditioned on $\mathsf{i\text{-}value}(\tau) \neq \bot$ is computationally indistinguishable from the distribution (τ', y') conditioned on $\mathsf{i\text{-}value}(\tau') \neq \bot$, where τ is C's view when interacting with $L(x, y)$ and R and $(\tau', y') \leftarrow \widehat{C}(x)$. (Note that the fact that a protocol is extractable does not imply immediately that it is strongly extractable: it may be the case that τ is indistinguishable from τ' but $\mathsf{i\text{-}value}(\tau)$ is in fact distinguishable from $\mathsf{i\text{-}value}(\tau')$ since $\mathsf{i\text{-}value}$ is not an efficiently computable function.) Note that in both variants, we allow the simulator to output an arbitrary value as its second output, if its first output τ' satisfies that $\mathsf{i\text{-}value}(\tau') = \bot$.

Since we only deal with the MIM setting from now on we will use the names *extractable* and *strongly extractable* and drop the qualifier "in the MIM setting". We say that Π is extractable (resp. strongly extractable) *with respect to non-synchronizing adversaries* if the extractibility condition (resp. strong extractibility condition) holds only against MIM adversaries C that use the *synchronizing* scheduling.

Relationship with non-malleability definition. As mentioned above, the condition of extractiblity is stronger than the notion of non-malleability as defined originally in [DDN91]. Dolev *et al.*defined a non-malleable protocol as a protocol where an MIM adversary cannot succeed in causing a non-trivial[32] relation between the left party's input and the intended value of the right session, more than can a simulator that only receives the public information. An extractable protocol is non-malleable because its second output will hit the relation with probability at most negligibly smaller than the adversary's. We note that latter works (such as [DDO+01] in the common reference string model) chose to use the condition of extractiblity as the *definition* of non-malleability, since this stronger condition is often needed in applications.

5.2 Constructing Extractable Protocols

Our approach to constructing an extractable zero-knowledge scheme and a strongly extractable commitment scheme is the following:

1. Reduce the problem to constructing only an extractable zero-knowledge scheme.
2. Show that our non-malleable coin-tossing protocol allows us to reduce the problem to the problem of constructing such a scheme in the *modified shared random string model.*
3. Give a simple construction of an extractable zero-knowledge scheme in the modified shared random string model
4. Show (in Section 6) that for both commitment and zero-knowledge schemes, one can transform an extractable protocol secure against synchronizing adversaries into a protocol secure also against non-synchronizing adversaries.

The following lemma handles Step 1 of this outline. That is, it says that in order to obtain both extractable commitments and zero-knowledge schemes, it is enough to construct the latter. We state and prove the lemma only for the case of syncrhonizing adversaries since this is the version we need in this section.

[32]In this context, we'll say that a relation R is non-trivial if $(x, x) \notin R$ for every x. This condition is required to rule out the trivial adversary that uses the relaying strategy in order to hit the relation.

Lemma 5.2. *If there exists a standard (non-interactive) perfectly binding commitment scheme* Com *and a zero-knowledge argument that is extractable w.r.t. synchronizing adversaries then there exists a commitment scheme that is strongly extractable w.r.t. synchronizing adversaries.*

Proof Sketch: To commit to a value y, run the following protocol:

Step L1 Left sends $\mathsf{Com}(y)$.

Step L,R2.x Left proves to right knowledge of the committed value using an extractable zero-knowledge argument.

For a synchronizing adversary C, denote by $\tilde{\alpha} = \mathsf{Com}(\tilde{y})$ the random variable representing C's first message in the right session. Suppose that use the simulator of the zero-knowledge scheme to simulate the proof of Steps L,R2.x (but still commit to y in Step L1). Since C is synchronizing, the distribution of $\tilde{\alpha}$ is unchanged. Yet, now we are able to extract $\tilde{y}$.

Now, suppose that we use $\mathsf{Com}(0^n)$ instead of $\mathsf{Com}(y)$ in the first step. No noticeable change should occur in the joint distribution of the transcript and $\tilde{y}$, since otherwise we would contradict the semantic security of the commitment scheme Com. Therefore our simulator will use a commitment to 0^n in the first step, and the simulator for the zero-knowledge argument in the second step.

□

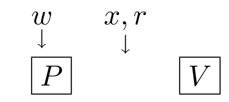
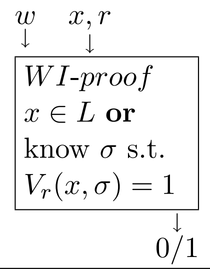
Note that the commitment scheme constructed has the property that the first message sent is from the sender to the receiver and that this message completely determines the committed value. We say that such a scheme has a *determining first message*.

5.3 Extractable Schemes in the Modified Shared Random String Model

We now define extractable and strongly extractable protocols in the *modified shared random string model*. The only difference is that we assume that $(r^{(1)}, aux^{(1)})$ and $(r^{(2)}, aux^{(2)})$ are generated independently by a generator algorithm (where r is random or pseudorandom), and then C chooses whether $r^{(1)}$ or $r^{(2)}$ will be used in the right session. The simulator will get $aux^{(1)}, aux^{(2)}$ as an additional input. We note that the resulting definition for extractable zero-knowledge schemes is almost identical to the definition of (single theorem) non-malleable NIZK [DDO+01]. The following theorem says that a non-malleable coin-tossing algorithm can indeed be used to "compile" a protocol secure in the modified shared random string model to a protocol secure in the plain model. Again, we state and prove the theorem only for the case of synchronizing adversaries.

Theorem 5.3. *Let* Π *be a non-malleable coin-tossing protocol and let* Π_{Ref} *be a protocol with a reference string that is extractable (w.r.t. synchronizing adversaries) in the modified shared random string model. Let* $\Pi \circ \Pi_{\mathsf{Ref}}$ *be the composition of* Π *and* Π_{Ref} *using Construction 1.2, then* Π_{Ref} *is extractable (w.r.t. synchronizing adversaries) in the plain model.*

Proof. Let C be an adversary for the combined protocol $\Pi \circ \Pi_{\mathsf{Ref}}$. Suppose that C is synchronizing. Then we can separate C into C_1, C_2 where C_1 is the adversary's strategy for the first phase (coin-tossing) and C_2 is the adversary's strategy for the second phase (running Π). We will pick $(r^{(1)}, aux^{(1)})$ and $(r^{(2)}, aux^{(2)})$ using the generator and then simulate C_1 using the simulator of the coin-tossing protocol ,where we give it $r^{(1)}, r^{(2)}$ as input. Let s denote the simulated output. Now, we can look at C_2 with the state s hardwired into it as an adversary for the extractable protocol

Public input: $x \in \{0,1\}^n$ (statement to be proved is "$x \in L$"), $r \in \{0,1\}^{l(n)}$: the reference string **Prover's auxiliary input:** w (a witness that $x \in L$)	
Steps P,V1.x (WI Proof): Prover proves to verifier using its input w via a witness-indistinguishable (WI) proof/argument of knowledge that either $x \in L$ or it knows a signature on x w.r.t. r where r is considered to be the public-key of a fixed one-time length-n message-size signature scheme (G, S, V). Verifier accepts if proof is completed successfully.	 w x, r WI-$proof$ $x \in L$ **or** know σ s.t. $V_r(x, \sigma) = 1$ 0/1

Protocol 5.4. An extractable zero-knowledge argument in the modified shared random string model

Π in the modified shared random string model and we can simulate it using the simulator Π and obtain (τ', y') such that $y' \in \textsf{i-value}(\tau')$. Yet τ' must be computationally indistinguishable from the output of C in an execution of $\Pi \circ \Pi_{\textsf{Ref}}$, and so we're done.

$\square$

5.4 An Extractable Zero-Knowledge Argument in the Modified Shared Random String Model

We now show a simple protocol for an extractable zero-knowledge argument in the modified shared random string.

Outline of the zero-knowledge argument. Our zero-knowledge argument is specified in Protocol 5.4. It has the following form: to prove that $x \in L$ when given a reference string r, the prover treats r as a verification key of some signature scheme, and proves (using a standard constant-round interactive zero-knowledge proof of knowledge)[33] that either $x \in L$ or that it knows a signature on x w.r.t. this verification key.

We see that our protocol requires a signature scheme that has a uniformly distributed public key. Fortunately, we only need a *one-time length restricted* signature scheme and so we can use the simple and well-known construction due to Lamport [Lam79], instantiating it with a one-way *permutation.*[34]

The theorem that we need to prove is the following:

Theorem 5.5. *Let (G, S, V) be a one-time length-n message signature scheme whose public key is a distributed uniformly in $\{0,1\}^{l(n)}$. Then, the instantiation of Protocol 5.4 with (G, S, V) is an extractable zero-knowledge argument in the modified shared random string model.*

Proof. Let C be an MIM adversary for Protocol 5.4. We construct a simulator $\widehat{C}$ for C in the following way:

[33] Actually, it is enough to use a witness indistinguishable argument of knowledge.

[34] That is, the public key is a list y_i^σ where $i \in [n]$ and $\sigma \in \{0,1\}$, and to sign a message $m = m_1, \ldots, m_n \in \{0,1\}^n$, one sends $x_i^{m_i} = f^{-1}(y_i^{m_i})$ for all $i \in [n]$, where $f(\cdot)$ is the one-way permutation.

1. Let $r^{(1)} = G(aux^{(1)})$ and $r^{(2)} = G(aux^{(2)})$ where G is the key generation algorithm of the signature scheme (G, S, V) and $aux^{(1)}, aux^{(2)}$ are two independently chosen random coin-tosses for G.

2. The simulator invokes $C(r^{(1)}, r^{(2)})$ and obtains a number $b \in \{1, 2\}$ such that if $b = 1$ then $r^{(1)}$ will be used as the reference string in the right session, and if $b = 2$ then $r^{(2)}$ will be used as this string.

3. The simulator computes a signature σ on x with respect to $r^{(1)}$ using $aux^{(1)}$. Using this signature σ, it construct a left-side strategy L_σ for the left side that uses the honest prover algorithm of the WI system to prove the true statement that either $x \in L$ or it knows a signature σ on x with respect to $r^{(1)}$.

4. The simulator simulates an interaction between L_σ, the adversary C, and the right side R. Let $\tau = (\tau_L, \tau_R)$ be the resulting view of the adversary, where τ_L (resp. τ_R) denotes the view of the adversary in the left (resp. right) session. We denote the statement proven by the adversary in the right session by $\tilde{x}$.

5. If in the view τ_R the verifier *accepted* the proof given by the adversary C, then the simulator will treat L_σ and C as a combined prover algorithm for the WI system and use the extractor of the WI system to either a witness $\tilde{y}$ that $\tilde{x} \in L$ or a signature $\tilde{\sigma}$ on $\tilde{x}$ w.r.t. $r^{(2)}$.[35]

6. The simulator outputs the view τ. In addition, if it obtained in the previous step a witness $\tilde{y}$ that $\tilde{x} \in L$ then it outputs this witness. Otherwise, its second output is $\perp$

By the WI condition, the simulator's first output is indeed indistinguishable from the view of C in a real interaction. Also, by the proof of knowledge condition, if the proof in the right session passes verification, then the simulator will indeed obtain either a witness $\tilde{y}$ for $\tilde{x}$ or a signature $\tilde{\sigma}$ on $\tilde{x}$ with respect to $r^{(2)}$. Thus, all that is left is to show that if $\tilde{x} \neq x$, then the second case (i.e., that the simulator obtains a signature) happens with negligible probability. To show this, one needs to observe that in Steps 4 and 5, the only secret information that the simulator uses is σ, which is a signature on x w.r.t. $r^{(1)}$. In particular, the simulator does *not* use either $aux^{(1)}$ or $aux^{(2)}$ in these steps. Therefore, if it managed to obtain a signature for any message $\tilde{x} \neq x$ w.r.t. $r^{(1)}$ in these steps that would contradict the existential unforgeability of the signature scheme (G, S, V) after seeing a single signature. If it managed to obtain a signature on *any* message (even on x) w.r.t. $r^{(2)}$ that would mean that it managed to forge even without seeing a single signature. $\square$

6 Handling Non-Synchronizing Adversaries

Throughout this paper, we have always concentrated on the case of adversaries that use the *synchronizing* scheduling. In this section, we justify this, by showing a way to transform protocols that are secure against synchronizing adversaries, into protocols that are secure against adversaries that may also use a *non-synchronizing* scheduling. (See Section 1.2.2 and Figure 3 for the definition of the synchronizing and non-synchronizing scheduling.)

[35]One needs to introduce a time-out mechanism in order to make sure this procedure runs in expected polynomial-time, in a similar way to [Lin01]. In fact, one can use directly a *witness-extended emulator* [BL02, Lin01] instead of a knowledge extractor. However, it seems that describing the simulator that uses the knowledge extractor is slightly simpler.

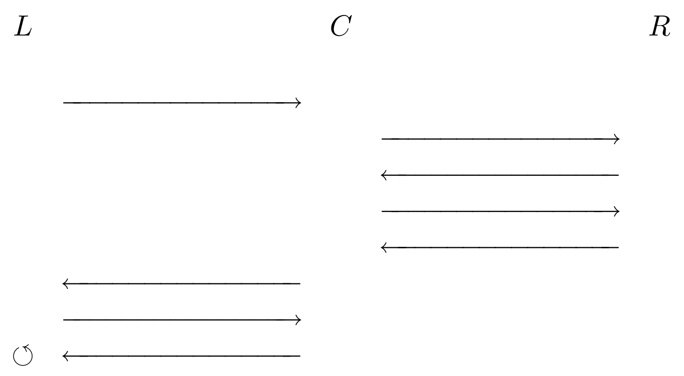


Figure 10: A non-syncrhonizing adversary: It is "safe" to rewind at the point marked by ↻.

Outline of our approach. The underlying observation behind the transformation of this section is that in some sense the synchronizing scheduling is the hardest schedule to deal with. To illustrate this, consider a malleable zero-knowledge proof of knowledge (P, V). We know that (P, V) is malleable, but it is still instructive to try to prove that it is non-malleable and see exactly in what place the proof fails. Suppose that we are trying to prove that (P, V) is extractable (in the sense of the previous section). The natural approach for a simulator would be to try to invoke the stand-alone simulator of (P, V) to simulate the left session, and the stand-alone extractor of (P, V) to extract a witness from the right session. The problem with this approach is that when simulating the left-session, one needs to combine both the adversary C and the right party R and to treat them as one single verifier algorithm. The problem is that when simulating a proof for this combined verifier, the simulator needs to *rewind* it in the process. This means that the simulator rewinds not only the adversary, but also the right party. As a consequence, when will fail when we try to run the knowledge extractor to extract the witness in a right session. Technically, the reason is that to run the extractor one needs to treat the simulator and the adversary as one combined prover algorithm for the proof system of the right session. However, because the simulator has the power to rewind the right party, it is not a "legitimate" interactive prover, and therefore the knowledge extractor cannot be applied to it.

We saw that we got intro trouble when we tried to simulate the left session and rewind the adversary, but needed to rewind the right party along with it. However, this is not necessarily the case if the adversary uses a *non-synchronizing* strategy. As can be seen in Figure 6, in this case there will be at least one point in the left session in which the adversary will return a message immediately to the left party, without any interaction with the right party (in Figure 6, this point is marked with a ↻). If it happens to be the point at which the simulator needs to rewind, then the simulator can rewind "safely" without having to rewind also the right party at the same time.

The question that remains is how to make sure that the point at which the adversary deviates from the synchronizing scheduling will be the same point in which the simulator needs to rewind. To do so, we follow the Richardson-Kilian [RK99] approach of *multiple rewinding opportunities* (See also [GL00, PRS02]).[36] That is, we will make essentially every point in the protocol a possible rewinding point. Therefore, whenever the adversary deviates from the synchronizing scheduling, we'll be able to use this to simulate it, and in fact have a simpler simulation, compared to the

[36]We note that this approach was already used implicitly in a context similar to ours by the earlier paper [DDN91].

simulators for synchronizing adversaries of the previous sections.

We now turn to the formal statement and proof of the theorem. We will start with the case of commitment schemes, and then describe how the proof can be modified for the case of zero-knowledge argument.

Theorem 6.1. *Suppose that there exists a constant-round strongly extractable commitment scheme with respect to synchronizing adversaries and that there exist perfectly hiding commitment schemes. Then, there exists a constant-round strongly extractable commitment scheme (with respect to adversaries that use arbitrary scheduling).*

6.1 Proof of Theorem 6.1

Let $\Pi_1 = (L, R)$ be a commitment scheme satisfying the conditions of the theorem. (That is, it is strongly extractable with respect to synchronizing adversaries.) We will prove Theorem 6.1 in the following way:

1. First, we will show a relatively simple construction of a commitment scheme Π_2 that is strongly extractable with respect to *non-synchronizing* adversaries. That is, for every MIM adversary C, there exist a 2-output simulator for C as long as C utilizes a *non-synchronizing* scheduling (but may fail if the adversary uses a synhcronizing scheduling).[37] The existence of such a scheme Π_2 lies behind our intuition that the synchronizing scheduling is the harder case to deal with. The construction of Π_2 will follow the Richardson-Kilian [RK99] paradigm of multiple rewinding points.

2. Secondly, we will construct a protocol Π which is a combination of Π_1 and Π_2. Roughly speaking, when committing to a string y, the first message of Π will be a two commitments to two random strings y_1 and y_2 such that $y_1 \oplus y_2 = y$. Then, we'll run Π_1 (committing to y_1) and Π_2 (committing to y_2) in parallel. That is, each message of Π will consist of one message from Π_1 and one message of Π_2.

3. We will then show that Π is a strongly extractable commitment scheme with respect to adversaries that utilize *any* scheduling strategy. Loosely speaking, for every adversary C, if C uses a synchronizing scheduling then we'll simulate it using the simulator of Π_1, and if C uses a non-synchronizing scheduling, then we'll simulate it using the simulator of Π_2.

Residual strategies. Before continuing with the proof, we recall the notion of *residual strategies* that will be useful for us later on. Recall that an interactive algorithm A is an algorithm that computes a *next-message function.* That is, given a random-tape r, and a list of messages $m_1, \ldots, m_i$, Algorithm A computes the next message m that it would send in an execution in which it has r as the random tape and received from the other party the messages $m_1, \ldots, m_i$. We use the notation $m = A(m_1, \ldots, m_i; r)$. If A is some interactive strategy and $v = \langle r, m_1, \ldots, m_j \rangle$ is a *partial view* of A, then the *residual strategy* of A with v fixed, is the next-message function that is obtained by "hardwiring" into A the view v. That is, this is the function that given a list of messages $m_{j+1}, \ldots, m_i$ (where $i \geq j + 1$) outputs $A(m_1, \ldots, m_j, m_{j+1}, \ldots, m_i; r)$. In the following sections, we will often consider an execution of an interactive algorithm A up to some point in a protocol, and then consider the residual algorithm A, which has the view up until that point fixed.

[37] Strictly speaking, the simulator will be slightly weaker than this, since we will allow it to fail also if the adversary utilizes an "almost synchronizing" scheduling, where this term will be defined below.

6.1.1 Construction of the scheme Π_2.

We now describe the construction of a commitment scheme Π_2 that will be secure with respect to *non-synchronizing* adversaries.

RK-iterations. To describe the scheme Π_2, we introduce a notion from [RK99], which we call an *RK-iteration.* An *RK-iteration* is a three round subprotocol α, β, γ, where the first and last messages are from the right party to the left party. Loosely speaking, we will define a condition under the left party is said to *win* an RK-iteration. The property that we require from an RK-iteration is that the probability that the left party wins in a normal interaction is negligible, *but* if the left party has the power to rewind the right party at the point β (i.e., get an additional reply γ' to some query $\beta' \neq \beta$) then it will be able to win with probability 1. To make this more concrete, we use the following implementation for an RK-iteration:

Let (G, S, V) denote a one-time length-restricted signature scheme. (Note that under the conditions of the theorem there exist one-way functions, and therefore there also exist such signature schemes [Lam79].) We use the following RK-iteration protocol:

Step R1 (Send VK): Right party runs $G(1^n)$ and obtains a verification and signing key-pair $\langle \alpha = VK, SK \rangle = \langle G_1(1^n), G_2(1^n) \rangle$. It sends the verification key, which we denote by α, to the left party.	$\xleftarrow{\alpha = G_1(1^n)}$
Step L2 (Send message): Left party chooses a message $\beta \leftarrow_{\mathrm{R}} \{0,1\}^n$ and sends it to the right party.	$\xrightarrow{\beta \leftarrow_{\mathrm{R}} \{0,1\}^n}$
Step R3 (Send signature): Right party computes a signature γ on the message β using the signing key, and sends it to the left party. If the right party does not send a valid signature at this step then we consider it as aborting the protocol.	$\xleftarrow{\gamma = S_{SK}(\beta)}$

We say that the left party *wins* the iteration if it knows (i.e., can output on an auxiliary tape) a valid signature with respect to the verification key α on some message $\beta' \neq \beta$. Clearly, the probability that an efficient left party wins an RK-iteration is negligible.

Description of the scheme Π_2. Let c denote the number of left party messages in the scheme Π_1.[38] The scheme Π_2 consists of the following: to commit to a value y, the left party sends a standard non-interactive commitment to y, and then the left and right parties perform c RK-iterations. After this, the left party proves using a perfectly witness-indistinguishable proof of knowledge that either it knows the value it committed to, or that it won one of the RK-iterations. For completeness, we provide a more formal description in Protocol 6.2.

Notation. Consider the left-party messages of the scheme Π_2 (Protocol 6.2). Note that c of these messages consist of a second message β of some RK-iteration. We call these messages the *core* of

[38]Since we can always add "dummy" messages to Π_1, we assume without loss of generality that the left party sends the first and last message in Π_1 and so the number of rounds in Π_1 is $2c - 1$.

Public input: 1^n (the plaintext string that will be committed to is of length n) **Left's (Sender's) auxiliary input:** $y \in \{0,1\}^n$ (plaintext to be comitted to)	$\overset{w}{\downarrow} \quad \overset{x,r}{\downarrow}$ $\boxed{L} \qquad \boxed{R}$
Step L1 (Standard commitment): Left sends to right a commitment to y using a standard (possibly malleable) non-interactive commitment Com	$\xrightarrow{\mathsf{Com}(y)}$
Steps L,R$2 \cdots 2c+2$ (RK-iterations): Left and right perform c RK-iterations. Note that we can combine the first step of each iteration with the last step of the previous one.	Repeat c times: $\xleftarrow{\alpha}$ $\xrightarrow{\beta}$ $\xleftarrow{\gamma}$
Steps L,R$2c+3 \cdots 2c+5$ (WI Proof): Left proves to right using a perfect-WI argument of knowledge that it either knows the value y committed to in Step L1 or that it won one of the RK-iterations (i.e., for some RK-iteration $\langle \alpha, \beta, \gamma \rangle$ it knows a signature δ w.r.t. α on some message $\beta' \neq \beta$). Right accepts if proof is completed successfully.	Perfect $WIPOK$ know y **or** $\exists$ RK-iteration $\langle \alpha, \beta, \gamma \rangle$ s.t. know δ, β' s.t. $\beta' \neq \beta$ and $V_\alpha(\beta', \delta) = 1$

Protocol 6.2. Π_2: A commitment scheme strongly extractable with respect to *non-synchronizing* adversaries.

the scheme Π_2. We say that an MIM adversary C for Π_2 uses an *almost synchronizing* scheduling if it's scheduling is synchronized with respect to all messages in the core. The scheduling of the other messages may be unsynchronized. More formally, let us number all the messages of the protocol from 1 to $2c+5$ (and so the left-party messages get odd numbers and the right-party messages get even numbers). We see that the set of core messages consists of all odd numbers between 3 and $2c+1$. We say that a scheduling for Π_2 is *almost synchronizing* if for every i in the core, the i^{th} (left-party) message is sent in the *right* session before the $i+1^{th}$ (right-party) message is sent in the *left* session. We note that we only refer to the scheduling of messages sent by the adversary since we can always assume without loss of generality that the honest parties send their messages immediately after they receive the previous message from the adversary. If a schedule is *not* almost-synchronizing then there exists a smallest i in the core for which the $i+1^{th}$ (right-party) message is sent in the left session *before* the i^{th} (left-party) message is sent in the right session. We call this i the *first unsynchronized core message* and we denote the RK-iteration that this message is part of as the *first unsynchronized RK-iteration.*

We have the following proposition:

Proposition 6.3. *The Scheme Π_2 is strongly extractable with respect to any adversary that uses a scheduling that is* not *almost-synchronizing.*

Proof. Let C be an adversary for Π_2 that utilizes a scheduling that is *not* almost-synchronizing. To show that Π_2 is strongly extractable we need to exhibit a simulator S for C. For every $y \in \{0,1\}^n$, the simulator S needs to simulate both the view of the adversary and the distribution of the value committed to by the adversary in the right session, when the adversary interacts with $L(y)$ and R (where $L(y)$ denotes the strategy of the left party when committing to input y). However, the simulator does *not* get y as an input, but only 1^n. Initially, we will construct a simulator S' that gets as input a string $e = \mathsf{Com}(y)$ as an input. Later, we will show how to get rid of this assumption, and obtain a simulator S satisfying the desired properties.

This simulator S' will work in the following way. The simulator will run internally an execution between the adversary C and the parties L and R. Note that the simulator can send $e = \mathsf{Com}(y)$ as its first message, and does not need to know y until the WI proof stage of the execution. Therefore, it can simulate perfectly the left party's strategy until that stage.

Since the adversary is not using an almost synchronizing strategy, at some point in the execution there will be a first unsyncrhonized core message i, as defined above. This is some point in which the left party sent its message of step i (which is the second step β of some RK-iteration) and obtained from the adversary a response γ immediately, without any interaction between the adversary and the right party taking place between the left party's message and the adversary's response. At this point, the simulator will run the following experiment: it will rewind the adversary to the point just before the i^{th} message was sent by the left party, and query the adversary with a new random message β'. The simulator will repeat this until the adversary again responds immediately to some β' with some answer γ'. (The overall number of steps will be expected polynomial number of times, since with probability p the simulator runs in $\frac{1}{p}\text{poly}(n)$ steps.)

The simulator will then record β' and γ', and continue to run the execution. However, it will now use a different residual strategy for the left party. Instead of using L, which is the honest left strategy, the simulator will use the strategy L' which when it gets to the WI proof stage, does *not* use the knowledge of the committed string, but rather uses its knowledge of β' and γ' to prove that it "won" some RK iteration. The simulator will continue the execution, and the view τ of the adversary in this execution will be the simulator's first output. Note that this view τ is distributed identically to the view of an adversary in a real execution because of the perfect WI condition.

Recall that the simulator needs also to output the message committed in the right session of τ, the simulator will rewind the parties L', R and the adversary C to the point in τ just before the adversary makes its proof of knowledge to the right party. It will then combine the adversary with the residual left strategy L' to obtain a prover algorithm for this system. The simulator then uses this prover as input to the knowledge extractor of the system and obtains either the message committed to by the adversary or a signature w.r.t. a verification key of some RK-iteration in the right session, which is on a message *different* from the message β of that iteration. However, because no rewinding of the right party at any of the RK iterations is done, the probability of the latter event is negligible (or otherwise we would have a forging algorithm for the signature scheme) and therefore with very high probability the simulator obtains the committed string that it needs as its second output.

Note that the distribution on the pair of the simulator's outputs is statistically indistinguishable from the distribution of the adversary's view and adversary's committed value in a real execution. Therefore, this simulator S' satisfies the strong extractability condition. To obtain a simulator S that does *not* get a $e = \mathsf{Com}(y)$ as an additional input, we simply have $S(1^n) = S'(\mathsf{Com}(0^n))$. By the indistinguishability of the commitment scheme Com, the pair output by S will be computationally indistinguishable from the pair output by S'. $\square$

Remark 6.4. Note that the proof of Proposition 6.3 actually shows a somewhat stronger condition than the statement. Not only does the simulator computes a view and witness pair (τ, y), but it also computes the left strategy L' that when executed with the adversary yields the view τ. We will later use this property of the simulator.

6.1.2 Construction and analysis of the scheme Π.

Now that we have a scheme Π_1 that is secure against adversaries that utilize the *synchronizing* scheduling, and a scheme Π_2 that is secure against adversaries that utilize a *non-synchronizing* scheduling, we are ready to construct our scheme Π which will be secure against adversaries that use arbitrary scheduling.

Properties of the Scheme Π_1. We note that we may assume that the scheme Π_1 has a *determining first message*,[39] as one can convert a scheme without this property into a scheme with this property by adding an initial message in which the left party sends a standard commitment to the right party. We also note that, as an extractable commitment scheme, the scheme Π_1 satisfies a standalone extraction property (or equivalently, it is a commit-with-extract scheme). That is, if we consider a *standalone* (not man-in-the-middle) execution of the scheme Π_1, where the left party is possibly cheating, then we can both simulate such an execution and obtain the string committed to by the left party. This is because one can regard the cheating left party as a degenerate man-in-the-middle adversary that does not use the messages it sees in the left session in its interaction in the right session. Such a degenerate adversary can without loss of generality use any scheduling (and in particular the synchronizing scheduling) and therefore we can use the simulator for Π_1 to obtain a simulated transcript of the right session along with the value committed to in this transcript.

[39]Recall that this means that first message in the protocol determines the possible value for the commitment.

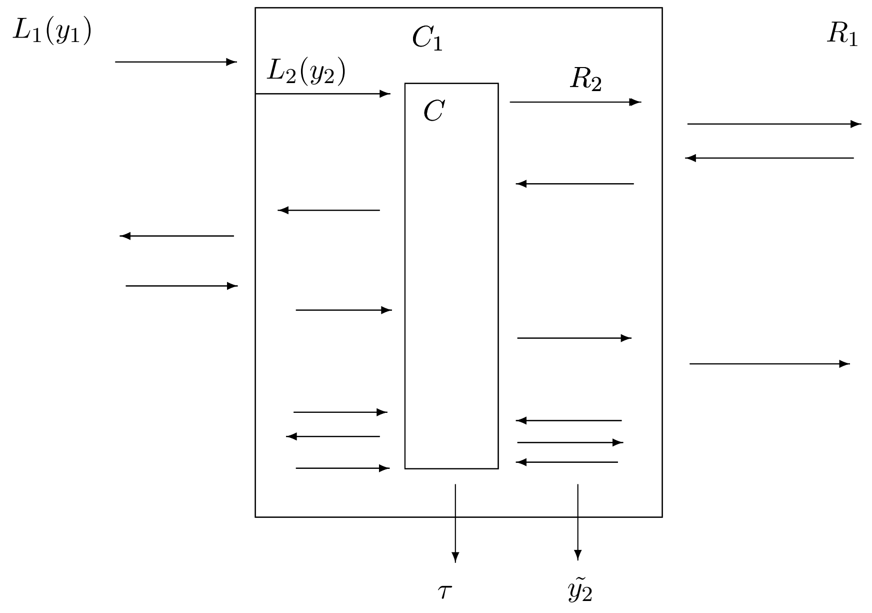


Figure 11: Converting adversary C to an adversary C_1 for the protocol Π_1.

The scheme Π. As outlined above, the combined scheme Π will be as follows:

1. The left and right parties run both protocols Π_1 and Π_2 in parallel to commit to y_1 and y_2 respectively. Note that Π_2 has more rounds than Π_1. We align the rounds of Π_1 to the *core* rounds of Π_2. That is, along with each of the c messages of the form β in rounds $3 \cdots 2c+1$ of Π_2, the left party sends a message that belongs to Π_1.

We claim that Π is a strongly extractable commitment scheme. Indeed, let C be an adversary for the scheme Π, we now describe a simulator S for C:

Sampling The simulator executes internally an execution of the protocol Π with C as a man-in-the-middle, where the left party commits to the value 0^n. The simulator then checks whether the scheduling of the adversary was almost synchronizing or not. That is, whether all messages in the core of Π_2 (and thus *all* messages of Π_1) are scheduled in a synchronized way.

The almost-synchronized case Suppose that in the sampled execution the adversary C used an almost synchronizing scheduling. In this case the simulator will use the simulator for Π_1. To do so, the simulator converts the adversary C into an adversary C_1 for the protocol Π_1. See Figure 11 for a schematic description of the adversary C_1. The adversary C_1 chooses a string $y_2 \leftarrow_{\mathrm{R}} \{0,1\}^n$, and runs an internal copy of C and an internal copy of the left strategy (committing to y_2) and right strategy for the protocol Π_2. When the adversary C_1 plays man-in-the-middle in the execution of Π_1 it forwards the messages it receives to its internal copy of C. Recall that since C is an adversary for Π, it also expects messages that belong to Π_2. To obtain these messages, the adversary C_1 uses its internal copy of the left and right strategy for Π_2. The adversary C_1 will be a *synchronizing* adversary for the protocol Π_1. This means that if its internal copy of C tries to schedule a message of Π_1 (or equivalently, a core message of Π) in a non-synchronized way then adversary C_1 will abort and output a special fail symbol. Otherwise, it will continue until the end of the protocol Π_1. Note that after the protocol Π_1 is finished, C_1's internal copy of adversary C will still not finish the last part of protocol Π (which consists of the proof of knowledge phase of Protocol Π_2). Adversary C_1 will simulate this part internally and use the knowledge extractor to obtain the string $\tilde{y}_2$ that C committed to in the right session.[40] The output of adversary C_1 is the view of its internal copy of the adversary C, along with the string $\tilde{y}_2$ committed by this internal adversary.

Note that if $y_1 = y \oplus y_2$ then (conditioned on being different from fail) the output of the adversary C_1 is identical to the view of the adversary C in an execution of Π. The simulator S now runs simulates C_1 using the simulator for Π_1 to obtain a pair $(\tau, \tilde{y}_2, \tilde{y}_1)$ where $\tau, \tilde{y}_2$ is computationally indistinguishable from C_1's output[41] and $\tilde{y}_1$ is the value committed by the adversary in the right session of the execution corresponding to τ. (If the simulator outputs a pair where $\tau = \mathsf{fail}$ then we repeat the experiment until we get a non-fail output.) Therefore, the pair $(\tau, \tilde{y}_1 oplus \tilde{y}_2)$ is computationally indistinguishable from the view and committed value of C in a real execution of Protocol Π.

The non-synchronizing case Suppose that in the sampled execution the adversary C did *not* use an almost-synchronized scheduling. In this case, we will use the simulator of Π_2 in an analogous way to the way we used the simulator of Π_1 in the previous step. That is, we will construct an adversary C_2 for the protocol Π_2 that has an internal copy of C inside it. The adversary C_2 will forward to C all the messages that it receives from the parties L_2 and R_2 in the protocol Π_1. In addition, it will execute internal copies of the honest left and right parties L_1 and R_1 of the protocol Π_1 (where it will provide L_1 with y_1 that is chosen at random in $\{0,1\}^n$). It will use these internal copies to supply the internal adversary C with the messages that it expects that come from the protocol Π_1. At the end of the protocol, the adversary C_2 outputs the view of its internal copy of C. As in a previous case, the output of C_2 after interacting with $L_2(y_2 \oplus y)$ and R_2 is identically distributed to the view of C after interacting with $L(y)$ and R. Therefore, if we simulate C_2, we obtain a pair of a transcript τ and the value $\tilde{y}_2$ committed to in the right session of Π_2 that indistinguishable from a transcript/value pair in a real execution of Π. The only thing that is missing is to obtain the value $\tilde{y}_1$ corresponding to this execution. However, this can be done using the fact that the simulator for protocol Π_2 constructed in the proof of Proposition 6.3 actually supplied us with a residual strategy for the left party L_2 that yields the transcript τ. Therefore, we can combine this left party L_2 with the adversary C to obtain a *standalone* adversary for the commitment scheme Π_1. Then, we can use the standalone extraction property of Π_1 to obtain the desired value $\tilde{y}_1$.

We see that both in the synchronized and unsynchronized case, the simulator S outputs a transcript/value pair that is indistinguishable from the transcript/value pair of a real execution of the adversary C. Intuitively, if we let p be the probability that C uses an almost synchronized

[40]Actually, the proof of knowledge may also yield a witness that C won one of the RK-iterations in the right session. However, because no rewinding of the right party is performed during the simulation, this can only happen with negligible probability.

[41]Formally, the simulator outputs a simulation of C_1's view, but C_1's output can be computed from its view.

scheduling, then the expected running time of the simulator will be $p\frac{1}{p}\text{poly}(n)+(1-p)\frac{1}{(1-p)}\text{poly}(n)$ which is equal to some polynomial $q(n)$ (Because, the simulator S will need to invoke the simulators for Π_1 and Π_2 a number of times that depends on $\frac{1}{p}$ until it gets another sample with the same property). However, because the probabilities in the simulation may have a negligible difference from the sampled probabilities, this is actually not the case, and we need to use the standard trick of [GK96] (see also [Lin01]) and introduce a timeout mechanism. That is, in the sampling stage, the simulator will actually sample many times an execution until it has an estimate $\tilde{p}$ for p that is within a factor 2 of p with probability $1-2^{-n^2}$ (this will mean that with probability p we sample $\frac{\text{poly}(n)}{p}$ number of times). The simulator will then use $T=\frac{4}{\tilde{p}}q(n)$ as a timeout value. That is, if the simulation takes more than T steps (which will happen with probability less than $\frac{1}{2}$) then the simulator, will restart from the beginning, where it will use at most n restarts. This will ensure expected polynomial-time simulation, and only introduce a negligible statistical bias.

□

6.2 The case of Zero-Knowledge

The analog for zero knowledge of Theorem 6.1 is the following:

Theorem 6.5. *Suppose that there exists a constant-round zero-knowledge proof for* **NP** *that is extractable with respect to synchronizing adversaries. Then, there exists a constant-round zero-knowledge proof for* **NP** *that is extractable with respect to adversaries that use arbitrary scheduling.*

The proof of Theorem 6.5 follows the proof of Theorem 6.1, and in fact is somewhat easier because we don't need to obtain a *strongly* extractable scheme in the case of zero-knowledge. As in the case of commitment schemes, we assume that there exists a c round zero-knowledge scheme Π_1 that is extractable w.r.t. synchronizing adversaries and we construct a $c+O(1)$-round zero knowledge proof system Π_2 that is extractable with respect to non-synchronizing adversaries. The scheme Π_2 will just be a variant of the [RK99] proof system, where when proving that some x is in some language M, the left and right party perform c RK-iterations and then the left party proves to the right party in WI that either $x \in M$ or that it won one of the RK-iterations. We'll combine the two schemes using a scheme Π. In the scheme Π the first message from the left party to the right party will be a string $y=\mathsf{Com}(b)$ for some $b \in \{1,2\}$. The left and right party will then run the schemes Π_1 and Π_2 in parallel, where the left will prove to the right in the scheme Π_b that either $x \in M$ or $y=\mathsf{Com}(b)$. We omit the full details and proof , since they are nearly identical to the case of commitment schemes. We note that one can also use the constructions of [DDN91] to reduce the problem of constructing a zero-knowledge scheme to the problem of constructing a commitment scheme and vice versa.

7 Conclusions and Open Questions.

We have shown a coin-tossing protocol that allows to transform many protocols secure in the shared random string model to protocols that are secure in the plain man-in-the-middle model. Using this coin-tossing protocol, we gave a constant-round non-malleable non-malleable commitment scheme and a constant-round non-malleable zero-knowledge argument system. It seems that our coin-tossing protocol can be applied in other settings in the man-in-the-middle model. In particular, it may be that it can be used to give a protocol for general 2-party secure computation in the man-in-the middle setting.

Complexity assumptions. Throughout this paper, we have used a subexponential hardness assumption (namely, that there exist some cryptographic primitives secure against 2^{n^ϵ}-sized circuits, where n is the security parameter). However note that we have *not* used the "complexity leveraging" technique [CGGM00] in this work, and hence in some sense this assumption is not inherent. The place where we used our assumption is in the construction of evasive sets, which we needed to be more efficient than the hardness assumption. However, in this construction the assumption was used for the purpose of derandomizing probabilistic machines. Thus it was sufficient to use any hardness bound $s(\cdot)$ such that under the assumption that $s(n)$-strong one-way functions exist, there is a pseudorandom generator mapping $o(\log s(n))$ bits to n bits (and hence one can derandomize polynomial-time algorithms in time $s(n)^{o(1)}$). Although any subexponential function satisfies this condition, one can use slower growing functions such as "half exponential" functions.[42] It is also known that for the purpose of derandomization it is sufficient to assume *worst-case* hardness (e.g., [NW88, IW97]). Thus, one can obtain the same result by assuming an exponential worst-case assumption (that implies that $\mathbf{BPP} = \mathbf{P}$) along with a "nice" super-polynomial hardness bound on the cryptographic primitives (e.g., $n^{\log n}$). An open question is to construct constant-round non-malleable protocols under the standard assumption of hardness against polynomial-sized circuits.

Strict polynomial time. The simulators presented in this work run in *expected* probabilistic polynomial-time. We believe that one can obtain simulators that run in *strict* probabilistic polynomial-time by using the commit-with-extract and proof of knowledge of [BL02]. However, this will result in (an even more) complicated non-black-box simulator, and thus we have chosen not to pursue this path.

Generalization to more parties. In this work we limited ourselves to a setting with two honest parties and one adversary. A natural question is whether these results generalize to a more general setting in which there are more honest parties communicating through an adversarial channel. The most general setting is when the number of honest parties is an arbitrary polynomial in the security parameter. A less general setting is when the number of honest parties is some fixed polynomial in the security parameter.

Another direction is to try to apply our techniques (diagonalization combined with universal arguments/CS proofs, non-black-box proofs of security) to other problems in cryptography.

Acknowledgements

I am most grateful to Alon Rosen. Alon introduced me to the problem of non-malleability and we had a lot of interesting and beneficial conversations on this topic and on this work. I would also like to thank Oded Goldreich for numerous discussions and insights and also for indispensable advice on the presentation. I also thank Ran Canetti, Yehuda Lindell, Tal Rabin and Salil Vadhan for helpful discussions.

[42] A function $f(\cdot)$ is half-exponential if $f(f(n)) > 2^n$.

References

[Bar01] B. Barak. How to go beyond the black-box simulation barrier. In *Proc. 42nd FOCS*, pages 106–115. IEEE, 2001. Preliminary full version available on `http://www.math.ias.edu/~boaz`.

[BG02] B. Barak and O. Goldreich. Universal Arguments and Their Applications. In *Annual IEEE Conference on Computational Complexity (CCC)*, volume 17, 2002. Preliminary full version available as Cryptology ePrint Archive, Report 2001/105.

[BL02] B. Barak and Y. Lindell. Strict Polynomial-Time in Simulation and Extraction. *SIAM Journal on Computing*, 33(4):783–818, Aug. 2004. Extended abstract appeared in STOC 2002.

[BOV03] B. Barak, S. J. Ong, and S. Vadhan. Derandomization in Cryptography, 2003.

[BG93] M. Bellare and O. Goldreich. On Defining Proofs of Knowledge. *Lecture Notes in Computer Science*, 740:390–420, 1993.

[Blu82] M. Blum. Coin Flipping by Phone. In *Proc. 24th IEEE Computer Conference (CompCon)*, pages 133–137, 1982. See also *SIGACT News*, Vol. 15, No. 1, 1983.

[BFM88] M. Blum, P. Feldman, and S. Micali. Non-Interactive Zero-Knowledge and Its Applications. In *Proc. 20th STOC*, pages 103–112. ACM, 1988.

[CF01] R. Canetti and M. Fischlin. Universally Composable Commitments. Report 2001/055, Cryptology ePrint Archive, July 2001. Extended abstract appeared in CRYPTO 2001.

[CGGM00] R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali. Resettable Zero-Knowledge. In *Proc. 32th STOC*, pages 235–244. ACM, 2000.

[CKPR01] R. Canetti, J. Kilian, E. Petrank, and A. Rosen. Black-Box Concurrent Zero-Knowledge Requires (Almost) Logarithmically Many Rounds. *SIAM Journal on Computing*, 32(1):1–47, Feb. 2003. Preliminary version in STOC '01.

[CR87] B. Chor and M. O. Rabin. Achieving independence in logarithmic number of rounds. In *Proc. 6th ACM PODC*, pages 260–268. ACM, 1987.

[DDO$^+$01] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust Non-interactive Zero Knowledge. In *CRYPTO ' 2001*, pages 566–598, 2001.

[DIO98] G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. Non-interactive and non-malleable commitment. In *Proc. 30th STOC*, pages 141–150. ACM, 1998.

[DKOS01] G. Di Crescenzo, J. Katz, R. Ostrovsky, and A. Smith. Efficient and Non-Interactive Non-Malleable Commitment. Report 2001/032, Cryptology ePrint Archive, Apr. 2001. Preliminary versoin in EUROCRYPT 2001.

[DDN91] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437 (electronic), 2000. Preliminary version in STOC 1991.

[DNS98] C. Dwork, M. Naor, and A. Sahai. Concurrent Zero Knowledge. In *Proc. 30th STOC*, pages 409–418. ACM, 1998.

[FLS99] Feige, Lapidot, and Shamir. Multiple Noninteractive Zero Knowledge Proofs Under General Assumptions. *SIAM J. Comput.*, 29, 1999.

[FFS87] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988. Preliminary version in STOC 1987.

[FS89] U. Feige and A. Shamir. Zero Knowledge Proofs of Knowledge in Two Rounds. In *Crypto '89*, pages 526–545, 1989. LNCS No. 435.

[FF00] M. Fischlin and R. Fischlin. Efficient Non-malleable Commitment Schemes. *Lecture Notes in Computer Science*, 1880:413–430, 2000. Extended abstract in CRYPTO' 2000.

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

[GK96] O. Goldreich and A. Kahan. How to Construct Constant-Round Zero-Knowledge Proof Systems for NP. *Journal of Cryptology*, 9(3):167–189, Summer 1996.

[GK92] O. Goldreich and H. Krawczyk. On Sparse Pseudorandom Ensembles. *Random Structures and Algorithms*, 3(2):163–174, 1992.

[GL00] O. Goldreich and Y. Lindell. Session-Key Generation using Human Passwords Only. Report 2000/057, Cryptology ePrint Archive, Nov. 2000. Extended abstract in CRYPTO 2001.

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989. Preliminary version in STOC' 85.

[IW97] R. Impagliazzo and A. Wigderson. $\mathbf{P} = \mathbf{BPP}$ if $\mathbf{E}$ Requires Exponential Circuits: Derandomizing the XOR Lemma. In *Proc. 29th STOC*, pages 220–229. ACM, 1997.

[Kil92] J. Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proc. 24th STOC*, pages 723–732. ACM, 1992.

[Lam79] L. Lamport. Constructing Digital Signatures from a One-Way Function. Technical Report CSL-98, SRI International, Oct. 1979.

[Lin01] Y. Lindell. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. In *Crypto '01*, pages 171–189, 2001. LNCS No. 2139.

[Lin03] Y. Lindell. A Simpler Construction of CCA2-Secure Public-Key Encryption Under General Assumptions. pages 241–254, 2003. LNCS No. 2656.

[Mic94] S. Micali. CS proofs. In *Proc. 35th FOCS*, pages 436–453. IEEE, 1994.

[Nao89] M. Naor. Bit Commitment Using Pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991. Preliminary version in CRYPTO' 89.

[NW88] N. Nisan and A. Wigderson. Hardness vs Randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, Oct. 1994. Preliminary version in FOCS' 88.

[PR05] R. Pass and A. Rosen. New and Improved Constructions of Non-Malleable Cryptographic Protocols. In *Proc.* 37*th STOC*. ACM, 2005.

[PRS02] M. Prabhakaran, A. Rosen, and A. Sahai. Concurrent Zero Knowledge with Logarithmic Round-Complexity. In *Proc.* 43*rd FOCS*. IEEE, 2002.

[RK99] R. Richardson and J. Kilian. On the Concurrent Composition of Zero-Knowledge Proofs. In *Eurocrypt '99*, pages 415–432, 1999. LNCS No. 1592.

[Sah99] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *Proc.* 40*th FOCS*, pages 543–553. IEEE, 1999.

[TW87] M. Tompa and H. Woll. Random Self-Reducibility and Zero-Knowledge Interactive Proofs of Possession of Information. In *Proc.* 28*th FOCS*, pages 472–482. IEEE, 1987.

A Universal Arguments

The following definition is reproduced from [BG02]:

We let $L_{\mathcal{U}} = \{(M,x,t) : \exists w \text{ s.t. } ((M,x,t),w) \in R_{\mathcal{U}}\}$, where $((M,x,t),w) \in R_{\mathcal{U}}$ if M accepts (x,w) within t steps. Let $T_M(x,w)$ denote the number of steps made by M on input (x,w); indeed, if $((M,x,t),w) \in R_{\mathcal{U}}$ then $T_M(x,w) \le t$. Recall that $|(M,x,t)| = O(|M| + |x| + \log t)$; that is, t is given in binary.

We consider a pair of (interactive) strategies, denoted (P,V), and let $(P(w),V)(y)$ denote the output of V when interacting with $P(w)$ on common input y, where $P(w)$ denotes the functionality of P when given auxiliary input w.

Definition A.1 (universal argument)**.** A *universal-argument system* is a pair of strategies, denoted (P,V), that satisfies the following properties:

Efficient verification: There exists a polynomial p such that for any $y = (M,x,t)$, the total time spent by the *(probabilistic)* verifier strategy V, on common input y, is at most $p(|y|)$. In particular, all messages exchanged in the protocol have length smaller than $p(|y|)$.

Completeness by a relatively-efficient prover: For every $((M,x,t),w)$ in $R_{\mathcal{U}}$,

$$\Pr[(P(w),V)(M,x,t) = 1] \;=\; 1$$

Furthermore, there exists a polynomial p such that the total time spent by $P(w)$, on common input (M,x,t), is at most $p(T_M(x,w)) \le p(t)$.

Computational Soundness: For every polynomial-size circuit family $\{\widetilde{P}_n\}_{n\in\mathbb{N}}$, and every $(M,x,t) \in \{0,1\}^n \setminus L_{\mathcal{U}}$,

$$\Pr[(\widetilde{P}_n,V)(M,x,t) = 1] \;<\; \mu(n)$$

where $\mu : \mathbb{N} \to [0,1]$ is a negligible function.

A weak Proof of Knowledge Property: For every positive polynomial p there exists a positive polynomial p' and a probabilistic polynomial-time oracle machine E such that the following holds:[43]

For every polynomial-size circuit family $\{\widetilde{P}_n\}_{n\in\mathbb{N}}$, and every sufficiently long $y = (M,x,t) \in \{0,1\}^*$ if $\Pr[(\widetilde{P}_n,V)(y) = 1] > 1/p(|y|)$ then

$$\Pr_r\left[\begin{array}{l} \exists w = w_1 \cdots w_t \in R_{\mathcal{U}}(y) \\ \qquad \forall i \in \{1,...,t\} \quad E_r^{\widetilde{P}_n}(y,i) = w_i \end{array}\right] \;>\; \frac{1}{p'(|y|)}$$

where $R_{\mathcal{U}}(y) \overset{def}{=} \{w : (y,w) \in R_{\mathcal{U}}\}$ and $E_r^{\widetilde{P}_n}(.,.)$ denotes the function defined by fixing the random-tape of E to equal r, and providing the resulting E_r with oracle access to $\widetilde{P}_n$.

The oracle machine E is called a *(knowledge) extractor*.

There are two differences between universal arguments and (interactive) CS-Proofs [Mic94]:

[43]Indeed, the polynomial p' as well as the (polynomial) running-time of E may depend on the polynomial p (which defines the noticeable threshold probability above).

1. The computational soundness in CS-Proofs needs to hold for cheating provers of size $poly(t)$. In contrast, the computational soundness in universal arguments needs only to hold for adversaries of size polynomial in the input length which is $|M| + |x| + \log(|t|)$.

2. We require that a universal argument satisfies a *proof of knowledge* condition. Note that we allow the knowledge extractor only polynomial time in the input length which is $|M| + |x| + \log(|t|)$, while the size of the witness may be t which is may be exponentially larger than the input length. Therefore, the knowledge extractor can output only an implicit representation of the witness.

We will use the following (almost trivial) lemma:

Lemma A.2. *Fix $f : \mathbb{N} \to \mathbb{N}$ to be some super-polynomial function (e.g., $f(n) = n^{\log n}$). Then, there exists an extractor algorithm E' that on input (M, x, t) and oracle access to a polynomial-sized circuit $\tilde{P}$ that such that $\Pr[(\widetilde{P}_n, V)(M, x, t) = 1] > \frac{1}{f(n)}$ runs in time $(f(n) \cdot t)^{O(1)}$ and outputs a witness $w \in R_{\mathcal{U}}(M, x, t)$ with probability $1 - \mu(n)$ for some negligible function μ.*

Proof Sketch: Using $t \cdot poly(n)$ steps, it is possible to convert an implicit representation of a witness into an explicit one. Thus by invoking the extractor E from the proof of knowledge property $n \cdot f(n)$ times and doing this conversion each time we can achieve the desired result. $\square$

We will use the following theorem:

Theorem A.3 ([BG02])**.** *Suppose that collision-resistent hash functions exist. Then, there exists a universal argument system. Furthermore, there exists such a system in which the prover is zero-knowledge.*

We stress that [BG02] only requires *standard* collision-resistent hash functions (i.e., secure against polynomial-sized circuits). However, it is not hard to see that if we assume that $2^{n^{\epsilon}}$-strong collision resistent hash functions exist, then (by modifying the security parameter accordingly) we can show that there exists a universal argument system in which the soundness, the proof of knowledge and zero-knowledge property hold against 2^{n^5}-sized circuits.