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

Boaz Barak[†]

Abstract

We present 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 (De Santis et. al., 2001) 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).*

1. Introduction

We consider the execution of two-party protocols in the so called "man-in-the-middle setting" (MIM). In this setting an adversary has complete control over the communication channel between two parties. This model has been studied by Dolev, Dwork and Naor [13] that defined a protocol to be *non-malleable* if it it is secure (in a specific sense) when executed in this setting.[1]

Dolev *et al* also gave protocols for a non-malleable commitment scheme, and a non-malleable zero-knowledge proof of knowledge. However, these protocols take an *unbounded* (at least logarithmic in the security parameter) number of rounds. No constant-round protocols was known for these problems in the *plain* model (i.e., without trusted parties or setup assumptions).

The situation is different in the *shared random string model* [6], where we assume that all parties have access to a common reference string that was chosen uniformly at random by some trusted dealer. In this model there exist 1-round (i.e., non-interactive) non-malleable commitment schemes [11, 12] and non-malleable zero-knowledge proof systems [26, 9].[2]

In this work we show how in some cases it is possible to convert protocols secure in the shared random string model, into protocols secure in the plain model (where there is no trusted party or setup assumptions). We do this by defining and constructing a constant-round *non-malleable coin-tossing protocol*. By combining our protocol with known protocols in the shared random string model (e.g., the non-malleable non-interactive zero-knowledge proof system of [9]) we obtain the first constant-round non-malleable commitment scheme and the first constant-round non-malleable zero-knowledge proof system.

Our techniques are different from the techniques used to construct previous protocols for 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.[3] Similar techniques

[*]The full version of this paper can be found on the Cryptology ePrint Archive or on `http://www.wisdom.weizmann.ac.il/~boaz`.

[†]Department of Computer Science, Weizmann Institute of Science, ISRAEL. Email: `boaz@wisdom.weizmann.ac.il`

[1]See below for more on the relation between the MIM model and the notion of non-malleability.

[2]See Section 1.3 for more on related works.

[3]The use of non-black-box reductions appears in the version of our protocol secure against *non-uniform* adversaries. Unfortunately, due to

were first used in [1] in the context of zero-knowledge proof systems. This work demonstrates that these techniques are applicable also in other settings in cryptography.

1.1. 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), there is a third party called 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 actually think in such a scenario of the protocol Π 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.

There are two extreme strategies that C can always use. 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 extreme strategy is the *blocking* strategy in which C plays its part in each session completely independent of the other session. Clearly, regardless of the protocol it is impossible to prevent the adversary from using one of these 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 these two extreme strategies (or such that it could not be advantageous to C to use any other strategy).

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 $\widetilde{\alpha}$.[4] 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.[5]

space considerations, this version of the protocol is only sketched here. A full description and analysis can be found in the full version [2].

Relation to non-malleability. Dolev *et al* [13] 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. 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 extreme 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* (in the sense of [13]).

The shared random string model. Consider two-party protocols where the two parties get as an additional input a string r called the *reference string*. We call such protocols *protocols with a reference string*. In the *shared random string model* [6], security is proved under the assumption that r was chosen uniformly at random by a trusted dealer D. If we combine the man-in-the-middle setting (MIM) with the shared random string model then we consider the following scenario. In the first phase, the trusted dealer D chooses a string r uniformly at random and provides it to all the parties (L,R, and C). In the second phase, the protocol is executed, as in the plain

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

[5]Of course one needs a computational version of independence.

MIM setting, in two concurrent sessions controlled by C. **Note:** In this work we are only interested in the MIM setting. Thus, from now on we will refer to an execution in the above scenario as an execution in the *shared random string model* (instead of the more cumbersome name "execution in the combined MIM and shared random string model").

Our aim in this paper is to convert two-party protocols for the man-in-the-middle setting that are secure in the shared random string model, to protocols secure in the plain MIM setting (where no trusted dealer exists) while incurring as small overhead as possible. Towards this end we will want to construct a two-party constant-round *coin-tossing protocol* for the MIM setting. A coin-tossing protocol is a protocol that lets two parties decide on a string r that should be a random (or at least pseudo-random) string.[6] Then, given a two-party protocol Π_{Ref} with a reference string that is secure in the shared random string model, one can convert it to a protocol Π_{Plain} in the plain model in the following way:

Construction 1.1. (Composition of 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).

Suppose that we execute the protocol Π_{Plain} 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). 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 modified shared random string model. 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 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.[7]

Intuitively, it seems that the ability to choose that the strings used in both sessions will be independent does 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. In fact we can prove that this is the case for all protocols for non-malleable commitments and non-malleable zero-knowledge proofs. 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.[8]

Definition 1.2 (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 1.2 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 $\widehat{C}$.

1.2. Our Results.

Our main result is the following:

[6]This only a minimal requirement, we will later see the actual requirements needed in order to satisfy our intended applications.

[7]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.

[8]Actually, for some applications (e.g., simulation of interactive protocols) it may be desirable to make a stronger requirements than Definition 1.2. Our protocol satisfies also these stronger requirements. However, we prefer to defer this discussion to the full version of this work [2].

Theorem 1.3. *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.*

In Section 2 we prove a slightly reduced version of Theorem 1.3. That is, we construct a non-malleable coin-tossing protocol that is secure only against adversaries that use a *uniform* algorithm and a *synchronizing* strategy. We sketch how this can rectified in Section 3.

Suppose that we're given a non-malleable coin-tossing protocol Π in the plain model and a non-malleable zero-knowledge protocol Π_{Ref} in the shared random string model (e.g., the non-malleable NIZK of [9]). It can be shown that if we compose Π and Π_{Ref} as in Construction 1.1, then we will get a non-malleable zero-knowledge argument in the *plain* model. Since a non-malleable zero-knowledge argument implies a non-malleable commitment scheme (the basic idea is to commit to a value and then use the zero-knowledge argument to prove knowledge of the committed value) we obtain:

Theorem 1.4. *Suppose that factoring integers is hard for 2^{n^ϵ}-sized circuits for some $\epsilon > 0$. Then:*

1. *There exists a constant-round non-malleable zero-knowledge argument system for* $\mathbf{NP}$.
2. *There exists a constant-round non-malleable commitment scheme that is statistically binding.*

Complexity assumptions. The assumptions we actually need is the existence of collision-resistent hash functions, trapdoor permutations, and dense cryptosystems [10] strong against 2^{n^ϵ}-sized circuits. The two latter conditions are inherited from [9]. Actually, our protocol can be proven secure under quantitatively weaker assumptions. That is, it is enough that the problems are hard for $T(n)$-sized circuits where $T(n)$ is a function such that *probabilistic* polynomial-time algorithms can be simulated by a $T(n)^{o(1)}$-time *deterministic* algorithms. For example this holds if the factoring problem is hard for $n^{\log n}$-sized circuits and there exists a problem in $\mathbf{E} = \mathbf{Dtime}(2^{O(n)})$ with *worst-case* circuit complexity $2^{\Omega(n)}$ [19]. We defer this analysis to the final full version of this paper.

Notions of non-malleability. We remark that the non-malleable commitment scheme we obtain is non-malleable with respect to committing (as the definition of [13], which stronger than the definition of [11], see [15]). In the current presentation we only show that our non-malleable zero-knowledge and commitment schemes are *liberal* non-malleable, in the sense that their simulators run in *expected* (as opposed to *strict*) polynomial-time. However, if we use as a component the zero-knowledge argument of [4], which has both strict polynomial-time simulator and knowledge extractor, we can obtain a strictly non-malleable commitment scheme and zero-knowledge argument system. This is the first such schemes in the plain model, as the schemes of [13] were only liberal non-malleable.[9] However, if we use the argument of [4], we will get a *non-black-box* simulator. We avoid doing so, for the sake of simplicity.

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 (**2**) "for every protocol that is secure in the modified shared random string model, its composition using Construction 1.1 with a non-malleable coin-tossing protocol yields a protocol that is secure in plain MIM setting" (i.e., it is non-malleable w.r.t itself). However this is problematic, not so much because Definition 1.2 is too weak, but mainly because the notion of "security" for protocols is not well-defined and depends on the particular application. We can however prove slightly more general statements than Theorem 1.4: see the full version [2] for more details.

Synchronizing adversaries. One can separate the strategy of C into two aspects. The first aspect is its *content* strategy; that is, how does C compute its messages. The second aspect is its *scheduling* strategy; that is, how does C choose when to send its 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} message in the right session after it receives the i^{th} message in the left session and vice versa (i.e., it sends the i^{th} message in the left session immediately after it received the i^{th} message in the right session). We call an adversary that always uses this scheduling a *synchronizing* adversary. Note that synchronizing adversaries can still use both the relaying and blocking *content* strategies (i.e., they can still either act transparently or independently in both sessions).

Perhaps surprisingly, it turns out that it is much easier to design protocols that are secure against non-synchronizing adversaries. In fact, in some sense the non-malleable commitment protocol of [13] (and also an earlier work in a different model by Chor and Rabin [8]) was designed to force the adversary to use a *non-synchronizing* strategy. In order to simplify the presentation, in this presentation we focus almost entirely on synchronizing adversaries (which is the hardest case to deal with). In Section 3.2 we sketch how our results can

[9]It may be that the zero-knowledge argument of [4] can also be used to modify the construction of [13] and obtain strict non-malleability.

be generalized to general (possibly non-synchronizing) adversaries.

1.3. Related work

The notion of non-malleability was defined by Dolev, Dowrk and Naor [13]. They also constructed a non-malleable encryption scheme, commitment scheme and a zero-knowledge proof system. Both the commitment schemes and the zero-knowledge proofs presented in their paper have a *non-constant* number of rounds. (In the plain model, where there are no trusted parties or setup assumptions, the schemes of [13] for non-malleable commitment and zero-knowledge have not been improved.)

More efficient protocols have been constructed in the *shared random string model* [6]. Sahai [26] constructed a single-round (i.e., *non-interactive*) non-malleable zero-knowledge proof system in this model. This scheme was improved by De Santis *et al* [9]. Di Crescenzo, Ishai and Ostrovsky [11] constructed in the shared random string model a non-interactive commitment scheme that is non-malleable in a weaker sense than [13] ("non-malleable w.r.t. opening" [15]). Di Crescenzo *et al* [12, Sec. 3] constructed in the shared random string model[10] a non-interactive commitment satisfying the stronger notion of non-malleability ("non-malleable w.r.t. committing"). Canetti and Fischlin [7] constructed in the shared random string model[11] 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 [7].

2. Construction of a Non-Malleable Coin-Tossing Protocol

In this section we present the construction of our protocol for non-malleable coin-tossing. Section 2.1 contains some notions and notations we use in the construction and analysis, and the cryptographic assumptions we make. In this section we will construct a protocol with two drawbacks: The first drawback 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 drawback is that our protocol will only be secure against adversaries that use the *synchronizing* scheduling. In Section 3 we sketch how we can modify this protocol to eliminate these drawbacks. It turns out that the first drawback is more serious than the second drawback.

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 the protocol of [21], except for the following modification. The modification is that while the protocol of [21] 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 [14] and has been used in several places since (e.g., [25, 1]). 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.[12] 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 1.2, 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 1.2 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.[13] 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 1.2.

2.1. Preliminaries

2.1.1 Notation

For a finite set $S \subseteq \{0,1\}^*$, we write $x \overset{\text{R}}{\leftarrow} S$ 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

[10] 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.

[11] Footnote 10 applies also here [22].

[12] This is the case also in [1], although there the simulator used the knowledge of the adversary's code instead of rewinding to ensure that Condition Y occurs.

[13] 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 the full version for details [2].

negligible function $\mu(\cdot)$.

2.1.2 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 $\epsilon > 0$).[14] 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 (we can obtain this by setting the security parameter for these primitives to be $n^{5/\epsilon}$). In contrast we aim to prove that our protocol is secure only against adversaries that use *uniform* probabilistic polynomial-time algorithms.[15]

2.1.3 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:[16]

Definition 2.1 (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 $\left|\Pr_{r \xleftarrow{R} R_n}[M(r)=1] - \Pr_{r \xleftarrow{R} \{0,1\}^n}[M(r)=1]\right| < \mu(n)$*

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 $r \in R_n$, $\Pr[M(r) \in R_n \setminus \{r\}] < \mu(n)$*

Sets with very similar properties have been shown to exist by Goldreich and Krawczyk [17]. Using similar methods we can prove

Theorem 2.2. *Suppose that 2^{n^ϵ}-strong one-way-functions exist, then there exists an evasive set.*

Proof Idea: Let f be a super-polynomial function such that $f(n) = 2^{o(n^\epsilon)}$. Observe that with high probability, a random subset of $\{0,1\}^n$ of size $f(n)$ will satisfy both the pseudorandomness (because $f(\cdot)$ is super-polynomial) and evasiveness (because $f(\cdot)$ is subexponential) properties. Under our assumptions, the choice of such a subset can be derandomized to obtain the constructibility property. See the full version [2] for details.

2.1.4 Cryptographic Primitives

Commitment Schemes. We let Comm denote a computational (i.e., perfectly binding) commitment scheme. That is, we denote a commitment to x by $\mathsf{Comm}(x) = \mathsf{Comm}(x; U_m)$. Note that we assume for simplicity that the commitment scheme Comm is non-interactive, as the scheme of Blum [5]. We can also use the 2-round scheme of Naor [24], that can be based on any one-way function.

Strong witness indistinguishable proofs of knowledge. We will use strong-witness-indistinguishable (SWI) proofs of knowledge (denoted $SWIPOK$), where SWI is as defined in [16, Sec. 4.6]. Since SWI is a weaker property than zero-knowledge we can always use a zero-knowledge proof of knowledge instead of a $SWIPOK$ system. We choose to use $SWIPOK$ to illustrate the places where we don't need the full power of zero-knowledge.[17] There are known constructions of $SWIPOK$ systems that use 4 rounds, under the existence of one-way functions [16, Sec. 4.6].

Universal arguments. Universal arguments were defined by Barak and Goldreich in [3]. They are a variant of (interactive) CS proofs [23, 20]. Loosely speaking, a *universal argument* is an interactive argument of knowledge for proving membership in $\mathbf{NEXP}$ (instead of $\mathbf{NP}$).[18] We will use a universal argument system that is also zero-knowledge ($ZKUARG$). There are known constructions of such systems that use 10 rounds under the existence of collision resistent hash functions [3].

[14]As mentioned in Section 1.2, our protocol can be proven secure under somewhat weaker assumptions at the cost of a more complicated analysis. We defer this analysis to the final full version of this paper.

[15]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 sketched in Section 3 is also in fact secure against 2^{n^δ}-sized circuits for some $\delta > 0$.

[16]For simplicity we "hardwired" into Definition 2.1 the constants and time-bounds required for our application.

[17]We will use $SWIPOK$ systems to prove knowledge of a committed value. The SWI property guarantees that the indistinguishability property of the commitment scheme is not harmed. The combination of a commitment and a $SWIPOK$ of knowledge of the committed value constitutes what was called a commit-with-extract scheme in [4].

[18]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 subexponential hardness assumptions; Although for simplification purposes we do use such an assumption in the current presentation, it is not inherent in our construction.

Step L1 (Commitment to r_1): Left party selects $r_1 \xleftarrow{\mathrm{R}} \{0,1\}^n$ and commits to it using a perfectly-binding commitment scheme. The commitment is denoted α_1.

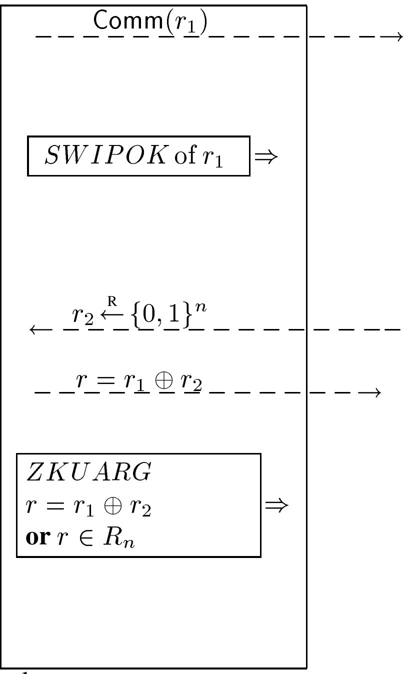
Steps L2.2-L2.4, R2.1-R2.3 (Prove knowledge of r_1): The left party proves to the right party its knowledge of the string r_1 using a $SWIPOK$. Specifically, the left party proves that it knows a pair of strings $s, r_1 \in \{0,1\}^n$ so that α_1 is a commitment to r_1 using s as randomness (in other words, α_1 equals $\mathsf{Comm}_s(r_1)$).

Step R3 (Send r_2): The right party selects a string $r_2 \xleftarrow{\mathrm{R}} \{0,1\}^n$ and sends it.

Step L4 (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 α_1.[19])

Steps L5.1-L5.9, R5.2-R5.10 (Prove that $r = r_1 \oplus r_2$): The left party proves, using a zero-knowledge universal-argument ($ZKUARG$), that either $r = r_1 \oplus r_2$ or $r \in R_n$.

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 2.3. *A non-malleable coin-tossing protocol for uniform adversaries*

2.2. The actual construction

Our non-malleable coin-tossing protocol is Protocol 2.3, described in the next page.

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 $r = r_1 \oplus r_2$. The reason that we need to use a universal argument (rather than a standard zero-knowledge proof or argument system for $\mathbf{NP}$) is that we are not guaranteed 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 sides 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 2.3 is a secure non-malleable coin-tossing protocol one needs to show that for any adversary C (that uses the *synchronizing* scheduling) there exists an ideal adversary $\widehat{C}$ that simulates C as is required in Definition 1.2. Indeed, let C be any probabilistic polynomial-time algorithm, and consider the execution of Protocol 2.3 in the man-in-the-middle setting where C plays the part of the channel. This execution is depicted in Figure 1. 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 as an arbitrary (efficiently computable) function of the previous messages.

[19]This is similar to the coin-tossing protocol of [21].

We need to simulate C by an "ideal" adversary $\widehat{C}$. The 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 2.3 is a secure non-malleable coin-tossing protocol it is enough to prove the following theorem:

Theorem 2.4. *Suppose that C is a probabilistic polynomial-time algorithm describing the strategy for a synchronizing adversary for Protocol 2.3, then there exists an algorithm $\widehat{C}'$ (computable by a probabilistic 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*

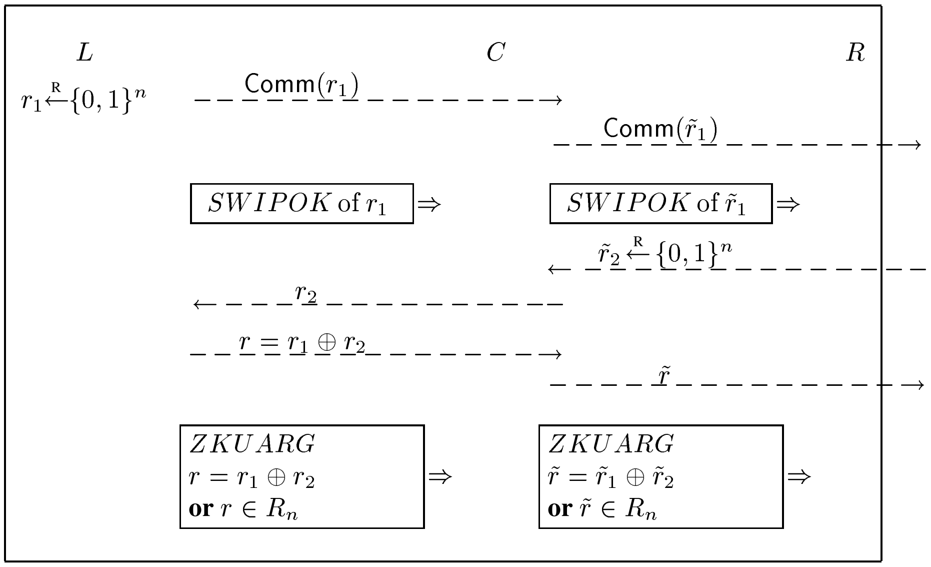

L
C
R
$r_1 \overset{R}{\leftarrow} \{0,1\}^n$
$\mathsf{Comm}(r_1)$
$\mathsf{Comm}(\tilde{r}_1)$
$SWIPOK$ of r_1
$SWIPOK$ of $\tilde{r}_1$
$\tilde{r}_2 \overset{R}{\leftarrow} \{0,1\}^n$
r_2
$r = r_1 \oplus r_2$
$\tilde{r}$
$ZKUARG$ $r = r_1 \oplus r_2$ **or** $r \in R_n$
$ZKUARG$ $\tilde{r} = \tilde{r}_1 \oplus \tilde{r}_2$ **or** $\tilde{r} \in R_n$

Figure 1. Execution of C in the man-in-the-middle setting

from the view of C in a real execution of Protocol 2.3 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 $\tilde{r} \in \{r^{(1)}, r^{(2)}\}$.*

Let C be a probabilistic polynomial-time algorithms representing the strategy of a synchronizing adversary for Protocol 2.3. In order to prove Theorem 2.4 we need to construct an algorithm $\widehat{C}'$ that simulates C. Such an algorithm $\widehat{C}'$ is depicted below in Figure 2. 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. Algorithm $\widehat{C}'$ starts by emulating C's execution by running the honest left and right strategy. Note that it does not need to use its inputs $r^{(1)}, r^{(2)}$ to do so. Algorithm $\widehat{C}'$ departs from the honest strategy in the right session (Steps L,R2.x) where it uses the *knowledge extractor* of the $SWIPOK$ to extract the string $\tilde{r}_1$.[20] Note that in order to do that, $\widehat{C}'$ needs to treat both C and the left party L as a single combined prover and rewind them both at the same time. Next, instead of letting $\tilde{r}_2$ be chosen uniformly 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 L4 $\widehat{C}'$ sends $r = r^{(1)}$ instead of $r = r_1 \oplus r_2$. Therefore with high probability the statement $r = r_1 \oplus r_2$ is false (and also $r \notin R_n$), but $\widehat{C}'$ uses the *simulator* for the $ZKUARG$ in order to simulate a proof for this false statement. Again it rewinds the honest right party while using the simulator. Note that all this can be carried out in expected polynomial-time.[21] 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 2.4. It turns out that Part 1 is not hard to prove and can be proved (quite easily) using a standard hybrid argument (relying on the security of the commitment scheme and zero-knowledge simulator). We will thus skip its proof here. 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.

2.3. Proof of Theorem 2.4 Part 2

From a first impression, by looking at Figure 2 one may think that Part 2 of Theorem 2.4 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 does not have access to the random coins of the verifier. But, since the algorithm $\widehat{C}'$ is using the simulator in the left ses-

[20] Actually what it needs to use is a *witness-extended emulator* [21], which is an algorithm that simulates the prover's view in addition to extracting a witness. However the existence of a knowledge extractor implies the existence of a witness-extended emulator [21].

[21] Actually we can also have a *strict* polynomial-time non-black-box simulator by using the protocols of [1] and [4].

sion, it will in actuality rewind also the verifier algorithm of the right party, thus ruining our ability to argue about the soundness of the universal-argument system. Indeed, this problem is real; if we 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) then such an adversary can ensure that $\tilde{r} = r$ (without the right party noticing anything wrong). We see that when we simulate such an adversary it will *not* be the case that $\tilde{r} = \tilde{r}_1 \oplus \tilde{r}_2$ (but rather $\tilde{r} = r^{(1)}$ which we indeed allow also). 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 3). Algorithm $\widehat{C}''$ behaves almost exactly as $\widehat{C}'$ with two differences:

1. In the left session (in Step L4) it chooses $r \overset{R}{\leftarrow} R_n$ instead of $r = r^{(1)}$. (This takes $2^{O(n^3)}$ steps using the constructibility property of R.)
2. Then, in the $ZKUARG$ it does not use the simulator but rather the honest prover algorithm (since the statement is true). (This takes $2^{O(n^3)}$ steps using the prover efficiency condition of the $ZKUARG$.)

We claim that the output of $\widehat{C}''$ is computationally indistinguishable 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: **(a)** $\tilde{r} \neq r$ **(b)** $\tilde{r} \neq r^{(2)} = \tilde{r}_1 \oplus \tilde{r}_2$ **(c)** 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[22] 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 $r \overset{R}{\leftarrow} R_n$. This means that we have a probabilistic polynomial-time algorithm that gets a string $r \overset{R}{\leftarrow} R_n$ 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. Filling the Gaps

In this section we give a rough sketch of the ideas how to extend Protocol 2.3 to obtain a protocol that is secure also against non-uniform adversaries and adversaries that may use non-synchronizing scheduling.

3.1. Non-uniform adversaries.

The actual construction and analysis of an extended version of Protocol 2.3 that is secure against non-uniform adversaries can be found in the full version of this work [1]. The extended protocol and its simulator are quite similar to Protocol 2.3 and its simulator. However, the proof of security involves an additional idea and uses the adversary's code in a *non-black-box* way.

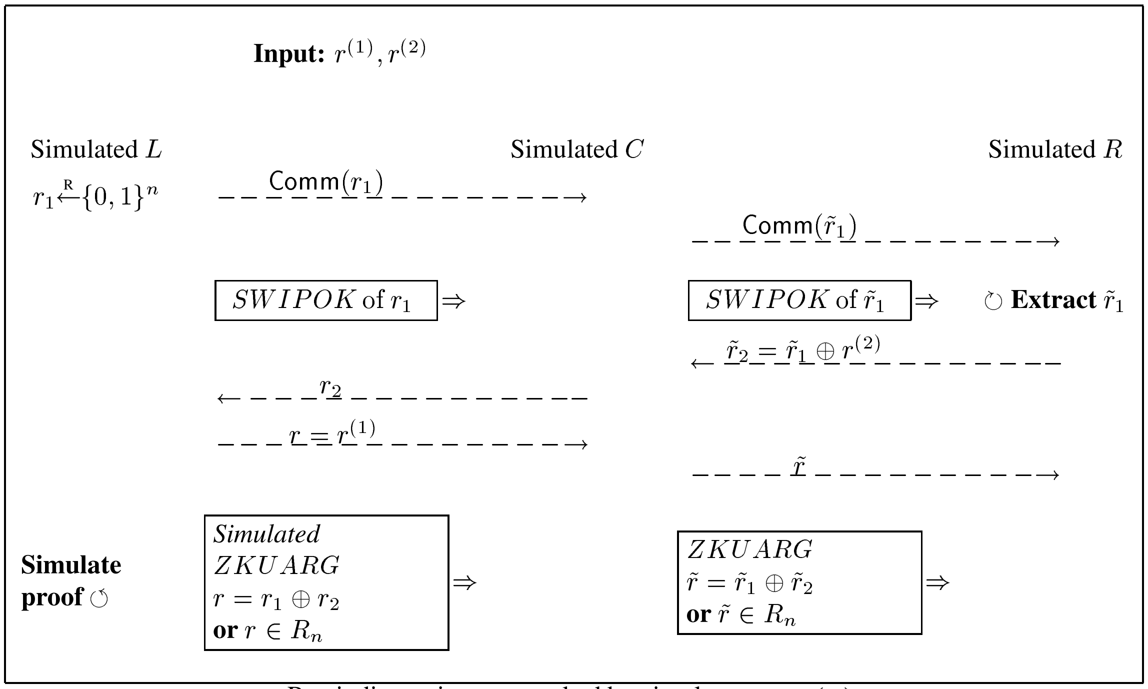
An important ingredient in the modified protocol is an extension of the notion of evasive sets. We define an *evasive set family* to be a family of subsets $\{R_\alpha\}_{\alpha \in \{0,1\}^*}$ (where for any α , $R_\alpha \subseteq \{0,1\}^*$) such that for any α, the set R_α is evasive with respect to probabilistic polynomial-time[23] algorithms that *get the string α as additional advice*. That is, the set R_α is both pseudorandom and 1-sample evasive with respect to such algorithms.

As mentioned above, our modified protocol will be very similar to Protocol 2.3, with an additional preliminary step (Step L,R0.x). In this step, the left party sends a commitment to a hashing value of some string α and the right party sends a commitment to a hash of some string β. In addition, each party proves knowledge of the string it hashed (i.e., the left party proves it knows the string α and the right party proves it knows the string β). The parties use a universal-argument for this step because we do not bound the length of α or β by any fixed polynomial in the security parameter. Another modification in the protocol is that in Step L,R5.x, the left party proves that $r = r_1 \oplus r_2$ or $r \in R_{\alpha \circ \beta} \cap \{0,1\}^n$ (instead of $R \cap \{0,1\}^n$ as in Protocol 2.3).

The prescribed left and right parties use $\alpha = \beta = 0^n$, and also our simulator will follow this strategy (i.e., use

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

[23] Actually, we consider $2^{O(n)}$-time algorithms instead of polynomial-time. However, this difference is not crucial and is mainly to simplify the exposition.


Input: $r^{(1)}, r^{(2)}$
Simulated L
Simulated C
Simulated R
$r_1 \overset{R}{\leftarrow} \{0,1\}^n$
$\mathsf{Comm}(r_1)$
$\mathsf{Comm}(\tilde{r}_1)$
$SWIPOK$ of r_1
$SWIPOK$ of $\tilde{r}_1$
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 $r \in R_n$
$ZKUARG$ $\tilde{r} = \tilde{r}_1 \oplus \tilde{r}_2$ or $\tilde{r} \in R_n$

Rewinding points are marked by circular arrows ↺, ↻

Figure 2. Algorithm $\widehat{C}'$ – simulation of C

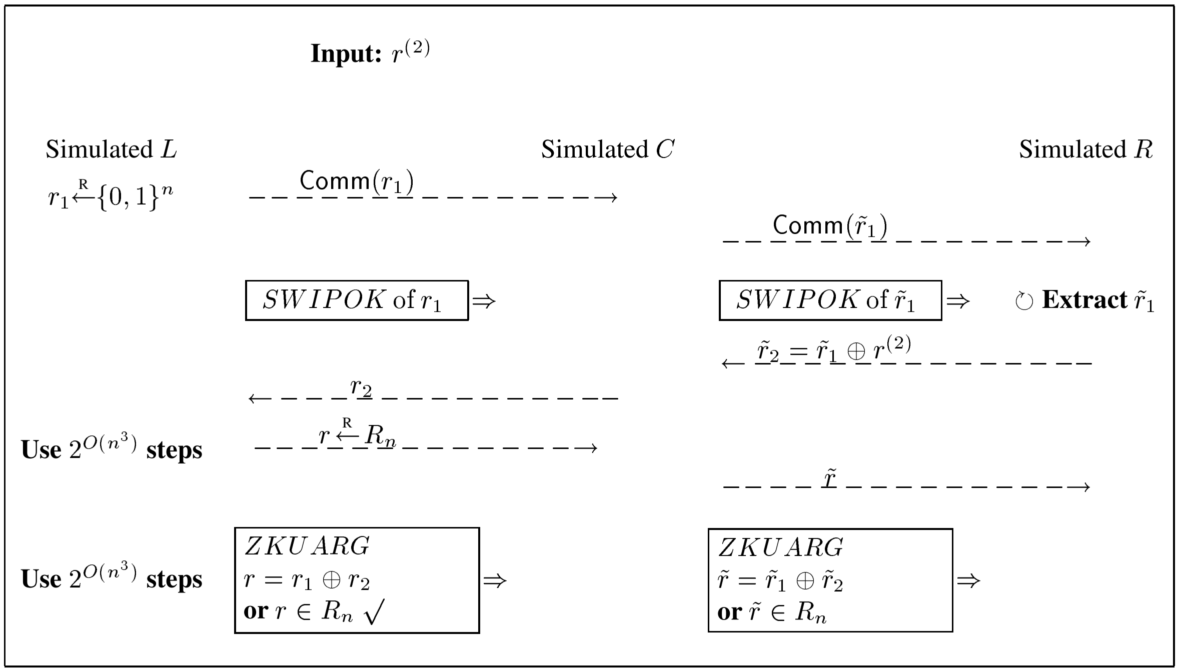

Input: $r^{(2)}$
Simulated L
Simulated C
Simulated R
$r_1 \overset{R}{\leftarrow} \{0,1\}^n$
$\mathsf{Comm}(r_1)$
$\mathsf{Comm}(\tilde{r}_1)$
$SWIPOK$ of r_1
$SWIPOK$ of $\tilde{r}_1$
Extract $\tilde{r}_1$
$\tilde{r}_2 = \tilde{r}_1 \oplus r^{(2)}$
r_2
Use $2^{O(n^3)}$ steps
$r \overset{R}{\leftarrow} R_n$
$\tilde{r}$
Use $2^{O(n^3)}$ steps
$ZKUARG$ $r = r_1 \oplus r_2$ or $r \in R_n$ $\surd$
$ZKUARG$ $\tilde{r} = \tilde{r}_1 \oplus \tilde{r}_2$ or $\tilde{r} \in R_n$

Figure 3. Algorithm $\widehat{C}''$

$\alpha = \beta = 0^n$). However, in order to prove a lemma analogous to Part 2 of Theorem 2.4 we present a $2^{O(n^3)}$-time algorithm (analogously to Algorithm $\widehat{C}''$ of Section 2.3) that uses $\alpha = \beta = \mathsf{desc}(C)$, where $\mathsf{desc}(C)$ denotes the description of the code of the adversary's algorithm. Note that this is a non-black-box use of the adversary's algorithm.

3.2. Non-synchronizing adversaries

Extending Protocol 2.3 to be secure against non-synchronizing adversaries is not too difficult. Recall that the main place in which we assumed the synchronizing strategy was in the construction of algorithm $\widehat{C}'$ (in the proof of Theorem 2.4). Algorithm $\widehat{C}'$ needed to apply both the knowledge extractor of the $SWIPOK$ in the right session and the simulator of the $ZKUARG$ in the left session. This may pose a problem if the adversary decides to change the scheduling so that the $ZKUARG$ of the left session and the $SWIPOK$ of the right session occur simultaneously. The solution is to use a $ZKUARG$ system with *multiple rewinding opportunities*. That is, we'll use a zero-knowledge universal argument where the simulator has several choices of the place to rewind the verifier. (We can obtain such an argument system using the ideas of Richardson and Kilian [25].) This is a trick that has been used before [25, 18, 13]. If the number of rewinding opportunities in the $ZKUARG$ is larger than the number of rounds that occur in our protocol until the end of the $SWIPOK$ phase is reached then there will always be an option to simulate the $ZKUARG$ in the left session without interfering with the extraction of the $SWIPOK$ system in the right session. There are a few more subtleties in handling non-synchronizing adversaries but all of them can be solved using the same trick.

Acknowledgments. 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.

References

[1] B. Barak. How to go beyond the black-box simulation barrier. In *42nd FOCS*, pages 106–115, 2001. Preliminary full version available on `http://www.wisdom.weizmann.ac.il/~boaz`.

[2] B. Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. Cryptology ePrint Archive, 2002. Full version of this work. Also available on `http://www.wisdom.weizmann.ac.il/~boaz`.

[3] B. Barak and O. Goldreich. Universal arguments and their applications. Cryptology ePrint Archive, Report 2001/105, 2001. Also posted as ECCC report TR01-093. Extended abstract to appear in CCC 2002.

[4] B. Barak and Y. Lindell. Strict polynomial-time in simulation and extraction. Cryptology ePrint Archive, Report 2002/043, 2002. `http://eprint.iacr.org/`. Extended abstract to appear in STOC 2002.

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

[6] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *Proceedings of 20th STOC*, pages 103–112, 2–4 May 1988.

[7] R. Canetti and M. Fischlin. Universally composable commitments. Report 2001/055, Cryptology ePrint Archive, July 2001. Extended abstract appeared in CRYPTO 2001.

[8] B. Chor and M. O. Rabin. Achieving independence in logarithmic number of rounds. In *Proceedings of 6th Annual ACM Symposium on Principles of Distributed Computing*, pages 260–268. ACM Press, Aug. 1987.

[9] 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.

[10] A. De Santis and G. Persiano. Zero-knowledge proofs of knowledge without interaction. In *33rd FOCS*, pages 427–436. IEEE Computer Society Press, 1992.

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

[12] 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.

[13] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437 (electronic), 2000. Preliminary version in 23rd STOC, 1991.

[14] Feige, Lapidot, and Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SICOMP: SIAM Journal on Computing*, 29, 1999.

[15] M. Fischlin and R. Fischlin. Efficient non-malleable commitment schemes. *Lecture Notes in Computer Science*, 1880:413–430, 2000. Extended abstract in CRYPTO' 2000.

[16] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.

[17] O. Goldreich and H. Krawczyk. On sparse pseudorandom ensembles. *Random Structures and Algorithms*, 3(2):163–174, 1992.

[18] 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.

[19] R. Impagliazzo and A. Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th STOC*, pages 220–229, 1997.

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

[21] Y. Lindell. Parallel coin-tossing and constant-round secure two-party computation. In *CRYPTO ' 2001*, pages 171–189, 2001.

[22] Y. Lindell. Personal communication, April 2002.

[23] S. Micali. CS proofs. In *Proceedings of 35th FOCS*, pages 436–453. IEEE Computer Society Press, 1994.

[24] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[25] R. Richardson and J. Kilian. On the concurrent composition of zero-knowledge proofs. In *EUROCRYPT: Proceedings of EUROCRYPT*, 1999.

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