Bounded-Concurrent Secure Two-Party Computation in a Constant Number of Rounds

Rafael Pass *
NADA
Royal Institute of Technology
SE-10044 Stockholm, Sweden
rafael@nada.kth.se

Alon Rosen †
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
alon@lcs.mit.edu

Abstract

We consider the problem of constructing a general protocol for secure two-party computation in a way that preserves security under concurrent composition. In our treatment, we focus on the case where an a-priori bound on the number of concurrent sessions is specified before the protocol is constructed (a.k.a. bounded concurrency). We make no set-up assumptions.

Lindell (STOC 2003) has shown that any protocol for bounded-concurrent secure two-party computation, whose security is established via black-box simulation, must have round complexity that is strictly larger *than the bound on the number of concurrent sessions. In this paper, we construct a (non black-box) protocol for realizing bounded-concurrent secure two-party computation in a* constant *number of rounds. The only previously known protocol for realizing the above task required more rounds than the pre-specified bound on the number of sessions (despite usage of non black-box simulation techniques).*

Our constructions rely on the existence of enhanced trapdoor permutations, as well as on the existence of hash functions that are collision-resistant against subexponential sized circuits.

1 Introduction

The task of *secure two-party computation* involves two parties that wish to evaluate a functionality $f(x, y) = (f_1(x, y), f_2(x, y))$ of their corresponding private inputs x and y. The evaluation process should supply the first party with the value $f_1(x, y)$ and the second party with the value $f_2(x, y)$ (one may also consider probabilistic functionalities in which case $f(x, y)$ is a random variable). Loosely speaking, the security requirement is that neither party learns more from the protocol other than the output, and that the output of each party is distributed according to the prescribed functionality. This should hold even in case that either one of the parties maliciously deviates from the protocol instructions. Shortly after its conceptualization, very strong results have been established for secure two-party computation. Specifically, it has been shown that *any* probabilistic polynomial-time computable two-party functionality can be securely computed, assuming the existence of enhanced trapdoor permutations [35, 22].

1.1 Secure Two-Party Computation in the Concurrent Setting

The original setting in which secure two-party protocols were investigated allowed the execution of a single instance of the protocol at a time (this is the so called *stand-alone* setting). A more realistic setting, however, is one which allows the *concurrent* execution of protocols. In the concurrent setting (originally introduced in the context of zero-knowledge [17, 15]), many two-party protocols are executed at the same time, involving many parties which may be talking with the same (or many) other parties simultaneously. This setting presents the new risk of a coordinated attack in which an adversary controls many parties, interleaving the executions of the protocols and choosing messages based on other partial executions of the protocol.

Unfortunately, security of a specific protocol in the stand-alone setting does not necessarily imply its security in the (more demanding) concurrent setting. It is thus of great relevance to examine whether the original feasibility results for two-party computation (in the stand-alone setting) still hold when many copies of the protocol are executed concurrently. Indeed, concurrent composition of cryptographic protocols has received a considerable amount of attention recently.

*Work partly done while visiting the Weizmann Institute of Science.
†Work done while at the Weizmann Institute of Science. Research supported in part by a grant from the Israel Science Foundation.

Many of the works that deal with protocol composition strongly rely on the existence of some kind of trusted set-up assumption. In some cases the usage of set-up assumptions seems to be necessary (e.g., Universal Composability [7, 12, 11]). However, it is not clear that for other types of protocol composition one cannot do without them (e.g., concurrent zero-knowledge [33, 24, 32], non-malleable string commitment [14, 2]).

In this paper we consider the problem of constructing a concurrently composable protocol for secure two-party computation. In contrast to most of the previous work in this area [7, 8, 12], and in accordance with recent work by Lindell [26], our goal is to construct a protocol that does not require *any set up assumptions*.

1.2 Previous Work

1.2.1 Composition Using Set-Up Assumptions

The literature discusses a wide array of set-up assumptions, most notably the timing assumption [15, 16], the Public-Key models [9, 13] and the Common Reference String (CRS) model [7, 12]. Whereas the former assumptions have been primarily used to obtain concurrent composition of zero-knowledge, the latter assumption has been also used to obtain strong composition theorems for secure two-party computation. Specifically, in the model assuming the existence of a Common Reference String, it has been shown how to achieve the very strong notion of Universal Composability (UC) for both two-party and multi-party secure computation [7, 12].

The universal composability setting (introduced by Canetti [7]) is even more demanding than the concurrent setting considered in this paper. It allows many sets of possibly different parties to run many protocols, and secure protocols may run concurrently with arbitrary other protocols. Security of a protocol in the UC setting would imply its security in the concurrent setting (as a special case). Unfortunately, the possibility of obtaining univeral composability without making any set-up assumptions (i.e., in the *plain model*) has been ruled out for a large class of functionalities [7, 8, 11]. This indicates that the notion of universal composabilty might be too strong to be realized in a meaningful way without making any set-up assumptions.

Recent work also seems to indicate that the CRS model might not reflect the security concerns that one would like to adress in reality [31]. Loosely speaking, standard simulation-based definitions of security in the CRS model do not cover certain "natural" security properties that are satisfied in the plain model (e.g., *deniability* of protocols). Moreover, known impossibility results for composition in the plain model (e.g., [10, 26]) will apply to any protocol that satisfies these "natural" security properties in the CRS model.

1.2.2 Concurrent Secure Two-Pary Computation

So far, the only treatment of concurrent two-party computation without set-up assumptions has been given by Lindell [26]. The definitional approach taken by Lindell is somewhat analogous to the approach taken by earlier works on concurrent zero-knowledge [15, 10]. Loosely speaking, the setting in which a two-party protocol should be proved to be secure involves a single (or many) honest parties that are running many concurrent executions of the same protocol. The honest parties are trying to protect themselves from a malicious adversary that controls a subset (or all) of the parties it is interacting with. Since it seems unrealistic (and certainly undesirable) for honest parties to coordinate their actions so that security is preserved, one must assume that in each instance of the protocol the honest party acts independently.

The conclusions reached by Lindell were mostly negative in nature. His main result is that any protocol for concurrent secure two-party computation, whose security is established via black-box simulation, must have round complexity that is strictly larger than the number of concurrent executions of the protocol. In particular, black-box simulation cannot be used in order to establish the security of two-party protocols when arbitrarily many protocols are allowed to run concurrently. Lindell's lower bound stands in sharp contrast to the case of zero-knowledge for $\mathcal{NP}$ where it has been shown that logarithmically many rounds are sufficient for black-box simulation in the concurrent setting [33, 24, 32].

Falling short of providing a concurrently composable protocol for secure two-party computation, Lindell resorts to the model of *bounded concurrency*. In the model of bounded concurrency, first considered by Barak [1] in the context of zero-knowledge, it is assumed that an a-priori bound on the number of concurrent sessions is specified before the protocol is constructed. As shown by Barak, it is possible to construct a constant-round bounded-concurrent zero-knowledge protocol for all languages in $\mathcal{NP}$ [1]. This goes well below previous impossibility results showing that logarithmically many rounds are necessary for black-box simulation of (even bounded) concurrent zero-knowledge [10]. Indeed, the security of Barak's protocol is proved using non black-box simulation techniques. Interestingly, it is still open whether it is at all possible to extend Barak's techniques to construct an *unbounded* concurrent zero-knowledge protocol (even using a super-constant number of rounds).

Using Barak's protocol, Lindell constructs a protocol for bounded concurrent secure two-party computation. The main feature required from Barak's protocol in the context of secure two-party computation is that its simulator does not make use of *rewinding* (a simulation technique inherent to black-box simulation [20, 25, 10]). In this context, the low round-complexity of Barak's protocol does not play a significant role.

Unfortunately, Lindell's protocol also relies on black-box simulation. As a consequence, the round-complexity of Lindell's protocol is severely limited by his own lower bound (on black-box simulation of concurrently composable secure two-party computation). Indeed, the number of rounds in the protocol is required to be strictly larger than the number of allowed concurrent sessions!

1.3 Our Results

Motivated by the above discussion, one might wonder whether Lindell's negative results are inherent to his definition of concurrent composition of two-party protocols (or even to any other "natural" definition of concurrency). Given the current state of knowledge, it is still conceivable that "non-trivial" concurrent composition of secure two-party protocols is beyond reach without resorting to some kind of set-up assumption (somewhat analogously to the situation in Universal Composability). In light of this, any protocol for (even bounded) concurrent two-party computation with "non-trivial" round-complexity would be interesting.[1]

In this work we address the above question and present a result of a positive nature. While our focus has been on feasibility, we also obtain the best one could hope for in terms of round-complexity. Specifically, we show how to construct a *constant-round* protocol for realizing bounded-concurrent secure two-party computation.

The security of our protocol relies on the existence of enhanced trapdoor permutations [18], as well as on the existence of strong collision resistant hash functions (i.e., so that for some $\kappa > 0$ forming collisions with probability greater than 2^{-n^κ} requires at least 2^{n^κ} time). For an integer m, the notion of m-bounded concurrent composition refers to a setting in which at most m copies of a protocol are executed concurrently (typically, $m = m(n)$ is a fixed polynomial in the security parameter). Our main theorem is stated below.

Main Theorem *Assume the existence of enhanced trapdoor permutations and the existence of strong collision-resistant hash functions. Then, for any two-party functionality f and for any integer m, there exists a protocol Π that securely computes f under m-bounded concurrent composition. Moreover, the number of communication rounds in Π is* constant.

[1] By "non-trivial" round complexity we mean that the number of rounds in the protocol is *strictly smaller* than the pre-specified bound on the number of sessions.

On the Cryptographic Assumptions Used. The above theorem relies on slightly strong, nevertheless standard, assumption on collision resistance of hash functions. As it turns out from our proof, this assumption can be somewhat relaxed to require the existence of both one-way functions that are hard to invert for sub-exponential sized circuits and of hash functions that are collision-resistant for quasi-polynomial sized circuits.

On the Model of Concurrency. The concurrent setting considered in this paper follows Lindell's treatment [26] which, in turn, is adapted from previous works on concurrent zero-knowledge. We consider a single (or many) honest parties that are running many concurrent executions of the same two-party protocol. The honest parties are trying to protect themselves from a malicious adversary that controls a subset (or all) of the parties it is interacting with. In each instance of the protocol the honest party is required to act independently. As observed by Lindell [26], this setting is equivalent to the case where many different pairs of parties run a protocol concurrently where each party is designated to be either a first party P_1 or a second party P_2 (this defines the role that it plays in the protocol execution). The adversary is then only allowed to corrupt a subset of the parties playing P_1 or a subset of the parties playing P_2; but cannot corrupt both parties P_1 and P_2 simultaneously. We further require that the adversary is *non-adaptive*, that is, the corrupted parties are chosen before the begining of the interaction.

1.4 Techniques

Our work relies in part on the works by Canetti, Lindell, Ostrovsky and Sahai [12] and Lindell [26]. The first of these works develops tools for proving security of two-party protocols in the framework of universal composability in the CRS model. The second work demonstrates how to use some of these tools in order to reduce the problem of concurrent composition of two-party protocols to the construction of special-purpose zero-knowledge protocols in the plain model. Once this reduction is provided, all that is required is to implement such protocols without resorting to any set-up assumption. Our first contribution is a conceptual simplification of Lindell's reduction. We do this by introducing a new type of adversary, called a *non-abusing* adversary. More details on such adversaries can be found in Sections 3.1.

The main technical ingredient of our work is the construction of two new special-purpose zero-knowledge protocols. These protocols satisfy the following extra properties:

Round efficiency: Both protocols have a constant number of rounds.

Bounded Concurrency: Both protocols retain their zero-knowledge property under bounded concurrent composition, even in presence of messages from other protocols.

Non-rewinding simulation: The simulators establishing the concurrent zero-knowledge property of the protocols do not use rewinding. Here we rely on Barak's non black-box simulation techniques [1]. In our setting, however, the simulator might simltaneously act as a prover in one protocol and as a verifier in another protocol. This introduces technical difficulties that Barak's protocol was not designed to deal with, and requires the usage of new ideas for performing the simulation.

Simulation Soundness: The protocols are simulation-sound with respect to each other. That is, the soundness of each one of the protocols is preserved even when the other protocol is simulated at the same time with the roles of the prover and verifier reversed. This introduces technical issues related to *non-malleability* [14]. The main problem here is to guarantee simulation-soundness for both protocols. Different solutions are required for each one of the protocols due to the symmetric nature of the problem. In one direction we use the idea of complexity leveraging (previously used in the context of *resettable zero-knowledge* and *simulation in quasi-polynomial time* [9, 30]). In the other direction, we introduce a new technique that enables us to prove simulation-soundness in a direct manner.

We feel that some of the techniques that we use for simulating the zero-knowledge protocols and for proving simulation-soundness might find application elsewhere. More details on our zero-knowledge protocols can be found in Section 3.2.

1.5 Future work

This paper essentially settles the question of bounded-concurrent composition of secure two-party computation. One issue that deserves improvement, however, is the strength of the underlying cryptographic assumptions.

Perhaps the most important open question in the context of concurrent composition is to establish whether it is possible to achieve unbounded concurrent composition of two-party protocols. This seems to require new ideas for improving currently known non black-box simulation techniques (due to Barak [1]). As a first step one could settle for super-constant round-complexity. We mention that, even in this relaxed case, nothing is known.

Another important question is whether one can achieve concurrently composable multi-party computation without an honest majority and without making any set-up assumptions. It seems that the techniques presented here do not suffice for achieving such a result.

1.6 Organization

Definitions of bounded-concurrent secure two-party computation can be found in Section 2. Section 3 contains a high level description of our proof, as well as a description of our new zero-knowledge protocols.

2 Definitions

In this section we present the definition for m-bounded concurrent secure two-party computation. Our definitions follows the ones of Lindell and we here present an abreviated version, taken almost verbatim from [26]. The basic description and definition of secure computation follows [23, 28, 5, 6]. We denote computational indistinguishability by $\equiv$, and the security parameter (and, for simplicity, the lengths of the parties' inputs) by n.

Two-party computation. A two-party protocol problem is cast by specifying a random process that maps pairs of inputs to pairs of outputs (one for each party). We refer to such a process as a functionality and denote it $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$, where $f = (f_1, f_2)$. That is, for every pair of inputs (x, y), the output-pair is a random variable $(f_1(x, y), f_2(x, y))$ ranging over pairs of strings. The first party (with input x) wishes to obtain $f_1(x, y)$ and the second party (with input y) wishes to obtain $f_2(x, y)$.

In the context of concurrent composition, each party receives a vector of inputs of polynomial length, and the aim of the parties is to jointly compute $f(x_i, y_i)$ for every i.

Adversarial behavior. In this work we consider a malicious, static adversary. That is, the adversary controls one of the parties (who is called corrupted) and may then interact with the honest party while arbitrarily deviating from the specified protocol. The focus of this work is not on fairness. We therefore present a definition where the adversary always receives its own output and can then decide when (if at all) the honest party will receive its output. The scheduling of message delivery is decided by the adversary.

Security of protocols (informal). The security of a protocol is analyzed by comparing what an adversary can do in the protocol to what it can do in an ideal scenario that is secure by definition. This is formalized by considering an *ideal* computation involving an incorruptible *trusted third party* to whom the parties send their inputs. The trusted party computes the functionality on the inputs and returns to each party its respective output. Unlike in the case of stand-alone computation, here the trusted party computes the functionality $p(n)$ times, each time upon different inputs. Loosely speaking, a protocol is secure if any adversary interacting in the real protocol (where no trusted third party exists) can do no more harm than if it was involved in the above-described ideal computation.

Concurrent execution in the ideal model. The ideal execution is defined as follows: The honest party sends all its inputs to the trusted party. The adversary thereafter sends its input in any order that it wishes to the trusted party, and directly receives the output. In particular, since we are considering a concurrent setting the adversary can schedule the order in which all protocols take place and choose its inputs adaptively. Finally, the adversary decides which outputs the honest party will recieve and sends a list of indexes to the trusted party, which in turn sends the output of these executions to the honest party. The honest party then outputs the results obtained from the trusted party, while the adversary can output an arbitrary function of its initial state and the messages obtained from the trusted party.

Let $p(n)$ be a polynomial in the security parameter, let $f : \{0,1\}^* \times \{0,1\}^* \mapsto \{0,1\}^* \times \{0,1\}^*$ be a functionality, and let S be a non-uniform probabilistic polynomial-time machine (representing the adversary). Then, the ideal execution of f (on input vectors $(\overline{x}, \overline{y})$ of length $p(n)$ and auxiliary input z to S), denoted $\text{IDEAL}_{f,\text{S}}(\overline{x}, \overline{y}, z)$, is defined as the output pair of the honest party and S in the above ideal execution.

We note that the definition of the ideal model does not include any reference to the bound m on the concurrency. This is because this bound is relevant only to the scheduling allowed to the adversary in the real model; see below.

Execution in the real model. We next consider the real model in which a real two-party protocol is executed (and there exists no trusted third party). Let $p(n)$ and $m = m(n)$ be polynomials, let f be as above and let Π be a two-party protocol for computing f. Furthermore, let A be a non-uniform probabilistic polynomial-time machine that controls either P_1 or P_2. Then, the real m-bounded concurrent execution of Π (on input vectors $(\overline{x}, \overline{y})$ of length $p(n)$ and auxiliary input z to A), denoted $\text{REAL}^m_{\Pi,\text{A}}(\overline{x}, \overline{y}, z)$, is defined as the output pair of the honest party and A, resulting from $p(n)$ executions of the protocol interaction, where the honest party always inputs its i^{th} input into the i^{th} execution.

The scheduling of all messages throughout the executions is controlled by the adversary. The bound on concurrency requires that the scheduling by the real model adversary must fulfill the following condition: for every execution i, from the time that the i^{th} execution begins until the time that it ends, messages from at most m different executions can be sent.

Security as emulation of a real execution in the ideal model. Having defined the ideal and real models, we can now define security of protocols. Loosely speaking, the definition asserts that a secure two-party protocol (in the real model) emulates the ideal model (in which a trusted party exists). This is formulated by saying that for every real-model adversary there exists an ideal model adversary that can simulate an execution of the secure real-model protocol.

Definition 1 (Security in the malicious model) *Let $m = m(n)$ be a polynomial and let f and Π be as above. Protocol Π is said to* securely compute f under m-bounded concurrent composition *if for every real-model non-uniform probabilistic polynomial-time adversary A controlling party P_i for $i \in \{1,2\}$, there exists an ideal-model non-uniform probabilistic polynomial-time adversary S controlling P_i, such that for every polynomial $p(n)$,*

$$\left\{\text{IDEAL}_{f,S}(\overline{x}, \overline{y}, z)\right\}_{n \in N} \equiv \left\{\text{REAL}^m_{\Pi,A}(\overline{x}, \overline{y}, z)\right\}_{n \in N}$$

where $\overline{x}, \overline{y} \in (\{0,1\}^n)^{p(n)}$ and $z \in \{0,1\}^*$.

Strengthening the definition. Definition 1 is rather simplistic in that it assumes that all inputs are fixed before any execution begins. We remark that our result also holds for a more general definition where inputs for later executions can depend on outputs of earlier executions.

3 Proof Outline

In this section we give a high-level outline of our proof, as well as a description of our new special-purpose zero-knowledge protocols. More details can be found in the full version.

The structure of our proof follows that of Lindell [26]. On a very high level, the proof can be divided into two parts. The first part shows how to reduce the problem of constructing a concurrently composable two-party protocol to the task of implementing two special-purpose zero-knowledge protocols. This part mainly follows [26] while introducing a new notion that conceptually simplifies the proof. It is described in Sections 3.1.

The second part involves the construction of the special-purpose zero-knowledge protocols. This part builds on Barak's bounded-concurrent zero-knowledge argument [1] as well as on new ideas related to non-malleability and concurrent simulatability. It is described in Section 3.2.

3.1 Reducing the Problem to Special-Purpose $\mathcal{ZK}$

3.1.1 Idealized Functionalities

Our starting point is the protocol of Canetti et al. [12]. In their paper, they show that the task of *concurrently composable* secure two-party computation can be achieved in a setting where the parties have access to a specific *idealized functionality*.[2] The notion of ideal functionalities is a central tool in the framework of universal composability [7], and can be thought of as the introduction of a trusted third party that is designed to perform a specific task. In the case of Canetti et al. [12], the protocol requires usage of the *ideal zero-knowledge proof of knowledge* functionality.

The IdealZK functionality. Informally, the ideal zero-knowledge proof of knowledge functionality (called IdealZK) for an $\mathcal{NP}$-relation R is specified as follows: The prover sends an instance-witness pair (x, w) to the ideal functionality which, in turn, sends $(x, 1)$ to the verifier if $(x, w) \in R$ and $(x, 0)$ otherwise.

The setting in which the parties have access to (multiple) ideal zero-knowledge functionalities is called the ZK-Hybrid model. Our motivation (as was also the motivation of Lindell [26]) is to remove the usage of the ideal zero-knowlegde functionality and implement a protocol that performs the same task but in the *real model*. If done in an appropriate way, this would give rise to a concurrently composable two-party computation protocol.

A natural (yet naïve) approach to implement such a protocol, would be to instantiate each idealized zero-knowledge call in the protocol of [12] with an invocation of a concurrent zero-knowledge argument. However, as pointed out in [26], this approach lacks in several aspects:

- The notion of concurrent zero-knowledge only considers the composition of the zero-knowledge protocol with itself. In the more complicated setting of secure two-party computation, it is essential that the protocol is zero-knowledge not only when being composed with itself, but also when being composed with other protocols running at the same time.

- In the setting of concurrent two-party computation, it is conceivable that a party acts as a prover in one session, while simultaneously acting as a verifier in a different session. (This problem does not occur in the setting of concurrent zero-knowledge, since each party is either a verifier or a prover.) Thus the problem of *malleability* arises: It is possible for an adversary to use messages from a proof that it is receiving in a proof that it is giving.[3]

In order to achieve the above mentioned task, we show a general transformation that transforms *any* protocol $\tilde{\Pi}$ in the ZK-Hybrid model into a protocol Π in the real model (without any ideal functionalities or set-up assumptions). This is what we call *realizing* the ZK-Hybrid model. Jumping ahead, we mention that this part also includes the construction of the special-purpose zero-knowledge protocols (as will be described in Section 3.2). We further show that the transformation satisfies the following properties:

Concurrency: The protocol's security is preserved under (bounded) concurrent composition.

Round-Efficiency: The protocol's round-complexity is preserved (up to a constant factor).

In particular the transformation can be applied to the protocol of [12] to obtain a bounded concurrent secure two-party computation protocol.

3.1.2 How to Realize Protocols in the ZK-Hybrid Model

The above transformation is performed in two steps:

1. Reduce the usage of the IdealZK functionality to the usage of a weaker (and easier to implement) ideal functionality. This functionality is named the WeakZK functionality.

2. Show how to transform any protocol using the WeakZK functionality into a protocol in the real model by "plugging in" the special-purpose protocols.

The WeakZK functionality. The first step consists of reducing the usage of IdealZK to the usage of WeakZK. (This reduction might be of independent interest as it in fact also is applicable in the multi-party setting withoutany bounds on concurrency). The latter functionality was introduced in [26] as a technical tool and is (informally) specified as follows: The prover sends a pair (x, b) to the ideal functionality which, in turn, sends (x, b) to the verifier if $x \in L$ and $(x, 0)$ otherwise (where $b \in \{0, 1\}$).[4]

The difference between the IdealZK and the WeakZK functionalities can be described as follows. While in the IdealZK functionality the prover needs to supply both a

[2] The result by Canetti et al. [12] is in fact stronger, as they construct a protocol that is universally composable [7]. Nevertheless, in this work we only require composition in the (less demanding) concurrent setting. One advantage arising from the usage of the protocol of [12] as a starting point is that it enables us to avoid many technicalities that arise when proving a general theorem on secure two-party computation from scratch.

[3] The situation in our case is even more complicated than in "traditional" treatments of non-malleability since, in our setting, the so-called man-in-the-middle is considered to succeed even if he convinces the verifier of the truthfulness of the *same* statement that he is being proven. In particular, known solutions for non-malleability do not apply here.

[4] In fact, our WeakZK functionality slightly differs from the one defined by Lindell. The reason we define it differently will become clear later when we introduce the new notion of *non-abusing* adversaries.

witness and a statment, in the WeakZK functionality it is sufficient for the prover to just supply a statement. In other words, the IdealZK functionality provides a *proof of knowledge*, while the ideal WeakZK functionality only provides a proof of validity. The extra bit b is used to let the prover "fail" in proving even true statements (clearly, one cannot force the prover to prove something if he does not want to do so). We call a model where parties have access to the WeakZK functionality, the WeakZK-Hybrid model.

Non-abusing adversaries. We note that the WeakZK functionality is not efficiently computable since, in general, it might be hard to decide whether a given instance belongs to an $\mathcal{NP}$ language without possessing a witness. This complicates the reduction from the usage of the IdealZK functionality to the usage of the WeakZK functionality. To overcome this problem, we introduce the notion of non-abusing adversaries. Non-abusing adversaries exist only in the WeakZK-Hybrid model. These adversaries (almost) always make "proper" usage of the WeakZK functionality. That is, they always explicitly tell the WeakZK functionality if the statement x is false by sending $(x, 0)$ to it. (We mention, however, that an non-abusing adversary may send $(x, 0)$ to the ideal functionality, even if $x \in L$). In other words, non-abusing adversaries only send values (x, b) that would make the WeakZK functionality send (x, b) to the verifier. Thus, for non-abusing adversaries the WeakZK functionality is efficiently computable.

Reducing between the hybrid models. Taking care of the above issue will enable us to show that any protocol $\tilde{\Pi}$ in the ZK-Hybrid model can be transformed into a protocol Π' in the WeakZK-Hybrid model, such that any *non-abusing* adversary in the WeakZK-Hybrid model can be simulated in the ZK-Hybrid model. This in particular means that, in the transformation between the two hybrid models, security is preserved. The proof of this statement makes use of a generic transformation of a zero-knowledge argument into a zero-knowledge argument *of knowledge* [4].

Now, in order to show how to realize any protocol in the ZK-Hybrid model, it remains to show how to transform any protocol Π' in the WeakZK-Hybrid model into a protocol Π in the real model. This should be done in a way than enables simulation of any adversary in the real model by a *non-abusing* adversary in the WeakZK-Hybrid model. Here we will also require the extra property that the simulator for the instantiated zero-knowledge proof does not make use of rewinding.[5] Jumping ahead, we mention that in order to show that the simulation can be done by a non-abusing adversary we will need to instantiate the calls to the WeakZK functionality by invocations of zero-knowledge protocols that are *simulation-sound* with respect to each other.

[5]To be accurate, some kind of rewinding is permissible in this setting (such rewinding has been used by Lindell [26]). However, one has to make sure that this rewinding does not rewind past messages that are *external* to the instantiated zero-knowledge protocols.

Completing the Reduction. The proof of the previous lemma thus boils down to the construction of two special-purpose zero-knowledge protocols satisfying the following requirements:

1. Both protocols compose concurrently with respect to arbitrary protocols.
2. Simulation of the protocols does not make use of rewinding.
3. The protocols are *simulation-sound* with respect to each other. That is, the soundness of each one of the protocols is preserved even when the other protocol is simulated at the same time with the role of the prover and verifier reversed.

3.2 The Special-Purpose $\mathcal{ZK}$ Protocols

Our special purpose zero-knowledge protocols are based on the bounded-concurrent $\mathcal{ZK}$ protocol of Barak [1]. Barak's protocol relies on the existence of strong collision resistant hash functions. Other tools used in the protocol are perfectly binding bit-commitments [29, 19] and a witness-indistinguishable universal argument (WI UARG)[3].

We start by giving a brief description of this protocol with an emphasis on a key property of its simulator. This property will enable us to perform simulation in our setting. More details on Barak's protocol can be found in [1]. The underlying idea behind Barak's protocol is the usage an $NTIME(n^{\log n})$ relation denoted R_{sim}.

Input: A triplet $\langle h, c, r\rangle$.

Witness: A program Π, a string $y \in \{0,1\}^{(|r|-n)}$, and a string s.

Relation: $R_{\mathsf{Sim}}(\langle h, c, r\rangle, \langle \Pi, s, y\rangle) = 1$ if and only if:

1. $c = \text{Com}\ (h(\Pi); s)$.
2. $\Pi(c, y) = r$ within $T(n)$ steps.

Figure 1. Barak's $NTIME(T(n))$ relation R_{sim}.

Let $T : \mathcal{N} \rightarrow \mathcal{N}$ be a "nice" function that satisfies $T(n) = n^{\omega(1)}$. Let $T' : \mathcal{N} \rightarrow \mathcal{N}$ be a function such that $T'(n) = T(n)^{\omega(1)}$. Suppose there exist a $T'(n)$-collision resistant hash functions ensemble $\{\mathcal{H}_n\}_{h\in\{0,1\}^n}$ where h maps $\{0,1\}^*$ to $\{0,1\}^n$. Barak's protocol is described in Figure 2 below.

Common Input: an instance $x \in \{0,1\}^n$ presumably in L, security parameter 1^n, length parameter $\ell(n)$.

Stage 1:

$V \to P$: Send $h \xleftarrow{R} \mathcal{H}_n$.

$P \to V$: Send $c = \text{Com}(0^n)$.

$V \to P$: Send $r \in \{0,1\}^{\ell(n)}$.

Stage 2:

$P \leftrightarrow V$: A *WI UARG* proving the OR of the following two statements:

1. $\exists\, w \in \{0,1\}^{\text{poly}(|x|)}$ so that $R_L(x,w) = 1$.
2. $\exists\, \langle \Pi, s\rangle$ so that $R_{\mathsf{Sim}}(\langle h, c, r\rangle, \langle \Pi, s\rangle) = 1$.

Figure 2. Barak's bounded-concurrent zero-knowledge argument for $\mathcal{NP}$.

As shown in [1], Barak's protocol is computationally sound. Moreover, given access to the verifier's code (or, equivalently, to the verifier's next message function), the protocol can be simulated without making use of rewinding. To simulate a specific session, the simulator commits to the verifier's next-message function (instead of committing to zeros). The verifier's next message function is then a program whose output depends on all the prover messages sent between messages c and r in the relevant session.

Now, if the length of all these messages is bounded by $\ell(n) - n$ the simulator will have a valid witness for stage 2 of the protocol. The zero-knowledge property then follows (with some work) from the witness indistinguishable property of stage 2.

Barak [1] has shown that the length of the prover's messages in a single execution can be bounded by $2n^2$. Thus, for m concurrent executions, taking $\ell(n) = 2m \cdot n^2 + n$ would do. We note that the length of the verifier's messages, *not counting r's*, can also be bounded by $2n^2$.

An important extra feature of the protocol is that it is in fact sufficient that the simulator has a "short" description of the prover's messages. The protocol is thus simulatable even if the total length of prover's messages is greater than $\ell(n) - n$, as long as the simulator has a description of a program for generating the prover's messages that is shorter than $\ell(n) - n$.

Our protocols. Let f be a one-way function with an efficiently recognizable range set, and let $\kappa > 0$ be so that f is not invertible by circuits of size 2^{n^κ}.[6] Let $k = 1/\kappa + 1$, $T'(n) = n^{\log^{k+1} n}$, and let $T(n)$ be a super-polynomial function such that $T(n) = T'(n)^{1/\omega(1)}$. Suppose there exist a $T'(n)$-collision resistant hash functions ensemble $\{\mathcal{H}_n\}_{h\in\{0,1\}^n}$ as required by Barak's protocol. We are ready to describe our special purpose protocols, $c\mathcal{ZK}_1$ and $c\mathcal{ZK}_2$ (depicted in Figures 3 and 4).

[6] The efficient recognizability requirement on f is not really necessary and it is introduced only for simplicity of presentation. In our case any one-way function with the above security properties would do.

Common Input: an instance $x \in \{0,1\}^n$ presumably in L, security parameter 1^n, length parameter $\ell(n)$.

Stage 1:

$V \to P$: Send $h \xleftarrow{R} \mathcal{H}_n$.

$P \to V$: Send $c_1 = \text{Com}(0^n)$.

$V \to P$: Send $r_1 \xleftarrow{R} \{0,1\}^{\ell(n)}$.

$P \to V$: Send $c_2 = \text{Com}(0^n)$.

$V \to P$: Send $r_2 \xleftarrow{R} \{0,1\}^{\ell(n)}$.

Stage 2:

$P \leftrightarrow V$: A *WI UARG*, that is sound against adversaries running in time $T'(n) = n^{log^{k+1} n}$, proving the OR of the following three statements:

1. $\exists\, w \in \{0,1\}^{\text{poly}(|x|)}$ so that $R_L(x,w) = 1$.
2. $\exists\, \langle \Pi, s\rangle$ so that $R_{\mathsf{Sim}}(\langle h, c_1, r_1\rangle, \langle \Pi, s\rangle) = 1$.
3. $\exists\, \langle \Pi, s\rangle$ so that $R_{\mathsf{Sim}}(\langle h, c_2, r_2\rangle, \langle \Pi, s\rangle) = 1$.

Figure 3. Our first special-purpose zero-knowledge protocol – $c\mathcal{ZK}_1$.

Simulating the protocols. The difference between $c\mathcal{ZK}_1$ and the protocol of Barak is that in $c\mathcal{ZK}_1$ the prover (simulator) is given two opportunities to guess the verifier's next message. We call each such opportunity a slot. Messages that are exchanged between the message c_i and the message r_i are said to be contained in slot i. Note that it is sufficient to have a "short" description of the messages contained in either slot 1 or slot 2 to succeed in the simulation of $c\mathcal{ZK}_1$.

$c\mathcal{ZK}_2$ is obtained by combining the quasi-polynomial time simulatable protocol of Pass [30] with the protocol of Barak. Note that, besides using Barak's technique to simulate in polynomial time, one can simulate $c\mathcal{ZK}_2$ by running in time $n^{O(\log^k n)}$ and inverting the one-way function by brute-force.

Let $\ell(n) = m(k \cdot 4n^2 + \texttt{length}(\Pi')) + n$, where Π' denotes the protocol in the WeakZK Hybrid model (that we wish to realize) and k is the total number of ideal zero-knowledge calls in one execution of Π'. We note that the total length of messages sent by a party, not including the r's in $c\mathcal{ZK}_1$, $c\mathcal{ZK}_2$, is bounded by $m(k \cdot 2n^2 + \texttt{length}(\Pi'))$, as both the length of the prover messages and verifier messages in $c\mathcal{ZK}_1$, $c\mathcal{ZK}_2$ is bounded by $2n^2$.

It follows that if the simulator can give a description of the various r's (each being of length $\ell(n)$) that is

Common Input: an instance $x \in \{0,1\}^n$ presumably in L, security parameter 1^n, length parameter $\ell(n)$.

Stage 1:

$V \to P$: Send $h \xleftarrow{R} \mathcal{H}_n$.

$V \to P$: Send $q = f(t)$, where $t \in^R \{0,1\}^{\log^k n}$. [7]

$P \to V$: Send $c = \mathrm{Com}(0^n)$.

$V \to P$: Send $r \xleftarrow{R} \{0,1\}^{\ell(n)}$.

Stage 2:

$P \leftrightarrow V$: A *WI UARG* proving the OR of the following three statements:

1. $\exists\, w \in \{0,1\}^{\mathrm{poly}(|x|)}$ so that $R_L(x,w) = 1$.
2. $\exists\, \langle \Pi, s\rangle$ so that $R_{\mathsf{Sim}}(\langle h,c,r\rangle, \langle \Pi, s\rangle) = 1$.
3. $\exists\, t'$ so that $q = f(t')$.

Figure 4. Our second special-purpose zero-knowledge protocol – $c\mathcal{ZK}_2$.

shorter than n^2 it will always succeed in the simulation of $c\mathcal{ZK}_1$,$c\mathcal{ZK}_2$. We hint that this is done by letting the simulator use a pseudorandom generator in order to generate verifier messages when playing the role of the verifier in other protocols.

Soundness in the stand-alone setting. By using a hash function that is secure against $T'(n)$-adversaries one can ensure that $c\mathcal{ZK}_1$ is sound against $T'(n)$ adversaries. It is straightforward to see that the functions $T(n)$ and $T'(n)$ satisfy the requirements needed to guarantee the soundness of the interactive argument.

As for $c\mathcal{ZK}_2$, soundness against polynomial time adversaries follows using the same argument as in [30], by combining the proof of knowledge property of the universal argument and the fact that the one-way function is hard to invert in sub-exponential time.

Simulation soundness. The last step is to guarantee that the soundness of each one of $c\mathcal{ZK}_1$,$c\mathcal{ZK}_2$ is not violated when simulating the other protocol with the roles of the prover and verifier reversed. To do this we show how to transform a cheating prover in this simulation/cheating scenario into a cheating prover for the stand alone case.

Suppose there exists a cheating prover P^* that manages to violate the soundness of one instance of protocol $c\mathcal{ZK}_i$ while it is verifying the simulation of multiple concurrent instances of the other $c\mathcal{ZK}$ protocol. We show how to construct a cheating prover P^{**} for a single instance of $c\mathcal{ZK}_i$ by forwarding the messages of this "cheating" instance of $c\mathcal{ZK}_i$ to an external honest verifier V (the instance in which P^* is actually cheating can be "guessed" by picking one of the instances of $c\mathcal{ZK}_i$ at random). V's replies are then forwarded by P^{**} back to P^* as if they were generated by the simulator. Since P^* is assumed to be cheating in this specific instance of $c\mathcal{ZK}_i$, and since the verifier messages used by P^{**} are actually the messages used by V, the (stand-alone) soundness of $c\mathcal{ZK}_i$ is presumably violated.

One final (subtle) issue that should be handled is that the code of V is not available to us. This means that the straightforward simulation of $c\mathcal{ZK}_i$ cannot be completed as it is (since it explicitly requires possession of a "short" explanation of the corresponding verifier messages). To overcome this problem we resort to alternative simulation techniques in each one of the protocols.

Simulating $c\mathcal{ZK}_2$ while verifying $c\mathcal{ZK}_1$: In this case we use the alternative simulator for $c\mathcal{ZK}_2$ (running in time $n^{O(\log^k n)}$). Note that this simulation technique does not require the verifier's code. Due to the WI property of the $UARG$, the presumed cheating prover will still succeed to convince an honest verifier of protocol $c\mathcal{ZK}_1$ with roughly the same probability. Since soundness of $c\mathcal{ZK}_1$ is guaranteed against provers running in time $n^{\log^{k+1} n}$ we derive a contradiction.

Simulating $c\mathcal{ZK}_1$ while verifying $c\mathcal{ZK}_2$: In this case we use the fact that it is sufficient to have a short description of the messages sent in either the first or the second slot of $c\mathcal{ZK}_1$ since the honest verifier's r message (to be "explained") can occur in at most one of the slots, polynomial time simulation is guaranteed. Again, the WI property of the $UARG$ guarantees that the the presumed cheating prover will still succeed to convince an honest verifier of protocol $c\mathcal{ZK}_2$ with roughly the same probability.

4 Acknowledgements

We are grateful to Yehuda Lindell for significant discussions on the subject and for giving us access to a manuscript of his work. We also thank Johan Håstad for valuable help and discussions. Thanks also to Oded Goldreich, Shafi Goldwasser, and Moni Naor for helpful conversations. The first author is grateful to the Weizmann Institute, and in particular Shafi Goldwasser, for letting him visit the Institute.

References

[1] B. Barak. How to go Beyond the Black-Box Simulation Barrier. In *42nd FOCS*, pages 106–115, 2001.

[7] To remove the efficient recognizability requirement from f, one could let V prove using a *Witness Hiding* proof that there exists an inverse to q under f.

[2] B. Barak. Constant-Round Coin-Tossing or Realizing the Shared-Random String Model. In *43rd FOCS*, pages 345-355, 2002.

[3] B. Barak and O. Goldreich. Universal Arguments and their Applications. *17th CCC*, pages 194–203, 2002.

[4] B. Barak and Y. Lindell. Strict Polynomial-Time in Simulation and Extraction. In *34th STOC*, pages 484–493, 2002.

[5] D. Beaver. Foundations of Secure Interactive Computing. In *CRYPTO'91*, Springer-Verlag (LNCS 576), pages 377–391, 1991. Repetition Lower the Error

[6] R. Canetti. Security and Composition of Multiparty Cryptographic Protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[7] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *34th STOC*, pages 494–503, 2002.

[8] R. Canetti and M. Fischlin. Universally Composable Commitments. In *Crypto2001*, Springer LNCS 2139, pages 19–40, 2001.

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

[10] R. Canetti, J. Kilian, E. Petrank and A. Rosen. Black-Box Concurrent Zero-Knowledge Requires (almost) Logarithmically Many Rounds. *SIAM Jour. on Computing*, Vol. 32(1), pages 1–47, 2002.

[11] R. Canetti, E. Kushilevitz and Y. Lindell. On the Limitations of Universally Composable Two-Party Computation Without Set-Up Assumptions. To appear in *EuroCrypt2003*.

[12] R. Canetti, Y. Lindell, R. Ostrovsky and A. Sahai. Universally Composable Two-Party and Multy-Party Computation. In *34th STOC*, pages 494–503,2002.

[13] I. Damgard. Eficient Concurrent Zero-Knowledge in the Auxiliary String Model. In *EuroCrypt2000*, LNCS 1807, pages 418–430, 2000.

[14] D. Dolev, C. Dwork and M. Naor. Non-Malleable Cryptography. *SIAM Jour. on Computing*, Vol. 30(2), pages 391–437, 2000.

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

[16] C. Dwork and A. Sahai. Concurrent Zero-Knowledge: Reducing the Need for Timing Constraints. In *Crypto98*, Springer LNCS 1462 , pages 442–457, 1998.

[17] U. Feige. Ph.D. thesis, Alternative Models for Zero Knowledge Interactive Proofs. Weizmann Institute of Science, 1990.

[18] O. Goldreich. Draft of a Chapter on general Protocols. Available at: `http://www.wisdom.weizmann.ac.il/~oded/PSBookFrag/prot.ps`

[19] O. Goldreich. *Foundation of Cryptography – Basic Tools*. Cambridge University Press, 2001.

[20] O. Goldreich and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *SIAM Jour. on Computing*, Vol. 25(1), pages 169–192, 1996.

[21] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *JACM*, Vol. 38(1), pp. 691–729, 1991.

[22] O. Goldreich, S. Micali and A. Wigderson. How to Play any Mental Game – A Completeness Theorem for Protocols with Honest Majority. In *19th STOC*, pages 218–229, 1987.

[23] S. Goldwasser and L. Levin. Fair Computation of General Functions in Presence of Immoral Majority. In *CRYPTO'90,* Springer-Verlag (LNCS 537), pages 77–93, 1990.

[24] J. Kilian and E. Petrank. Concurrent and Resettable Zero-Knowledge in Poly-logarithmic Rounds. In *33rd STOC*, pages 560–569, 2001.

[25] J. Kilian, E. Petrank and C. Rackoff. Lower Bounds for Zero-Knowledge on the Internet. In *39th FOCS*, pages 484–492, 1998.

[26] Y. Lindell. Bounded-Concurrent Secure Two-Party Computation Without Setup Assumptions. To appear in *34th STOC*, 2003.

[27] S. Micali. CS Proofs. *SIAM Jour. on Computing*, Vol. 30 (4), pages 1253–1298, 2000.

[28] S. Micali and P. Rogaway. Secure computation. Unpublished manuscript, 1992. Preliminary version in *CRYPTO'91*, Springer-Verlag (LNCS 576), pages 392–404, 1991.

[29] M. Naor. Bit Commitment using Pseudorandomness. *Jour. of Cryptology*, Vol. 4, pages 151–158, 1991.

[30] R. Pass. Simulation in Quasi-polynomial Time and its Application to Protocol Composition. In *EuroCrypt2003*, Springer LNCS 2656, pages 160–176, 2003.

[31] R. Pass. On Deniability in the Common Reference String and Random Oracle Model. To appear in *Crypto2003*

[32] M. Prabhakaran, A. Rosen and A. Sahai. Concurrent Zero-Knowledge with Logarithmic Round Complexity. *Proceedings of the 43rd annual IEEE symposium on Foundations of Computer Science (FOCS 2002)*, pages 366-375 2002.

[33] R. Richardson and J. Kilian. On the Concurrent Composition of Zero-Knowledge Proofs. In *EuroCrypt99*, Springer LNCS 1592, pages 415–431, 1999.

[34] A. Rosen. A note on the round-complexity of Concurrent Zero-Knowledge. In *Crypto2000*, Springer LNCS 1880, pages 451–468, 2000.

[35] A. Yao. How to Generate and Exchange Secrets. In *27th FOCS*, pages 162–167, 1986.