Concurrent Non-Malleable Zero Knowledge

Boaz Barak
Department of Computer Science
Princeton University
boaz@cs.princeton.edu

Manoj Prabhakaran
Department of Computer Science
University of Illinois at Urbana-Champaign
mmp@cs.uiuc.edu

Amit Sahai*
Department of Computer Science
University of California Los Angeles
sahai@cs.ucla.edu

Abstract

We provide the first construction of a concurrent and non-malleable zero knowledge argument for every language in **NP**. *We stress that our construction is in the plain model with no common random string, trusted parties, or super-polynomial simulation. That is, we construct a zero knowledge protocol* Π *such that for every polynomial-time adversary that can adaptively and concurrently schedule polynomially many executions of* Π*, and corrupt some of the verifiers and some of the provers in these sessions, there is a polynomial-time simulator that can simulate a transcript of the entire execution, along with the witnesses for all statements proven by a corrupt prover to an honest verifier.*

Our security model is the traditional model for concurrent zero knowledge, where the statements to be proven by the honest provers are fixed in advance and do not depend on the previous history (but can be correlated with each other); corrupted provers, of course, can chose the statements adaptively. We also prove that there exists some functionality $\mathcal{F}$ *(a combination of zero knowledge and oblivious transfer) such that it is impossible to obtain a concurrent non-malleable protocol for* $\mathcal{F}$ *in this model. Previous impossibility results for composable protocols ruled out existence of protocols for a wider class of functionalities (including zero knowledge!) but only if these protocols were required to remain secure when executed concurrently with arbitrarily chosen different protocols (Lindell, FOCS 2003) or if these protocols were required to remain secure when the honest parties' inputs in each execution are chosen adaptively based on the results of previous executions (Lindell, TCC 2004).*

We obtain an $\tilde{O}(n)$*-round protocol under the assumption that one-to-one one-way functions exist. This can be improved to* $\tilde{O}(k \log n)$ *rounds under the assumption that there exist* k*-round statistically hiding commitment schemes. Our protocol is a black-box zero knowledge protocol.*

*Research supported in part by an Alfred P. Sloan Foundation Research Fellowship, an Intel equipment grant, and NSF ITR/Cybertrust grants 0205594, 0456717 and 0627781.

1. Introduction

In the two decades since their introduction [17], zero-knowledge proofs have played a central role in the study of cryptographic protocols. Intuitively speaking, a zero-knowledge proof is an interactive protocol that allows one party (a "prover") to convince another party (a "verifier") that some statement is true, without revealing anything else to the verifier. The zero knowledge property was formalized in [17] by requiring that the verifier can efficiently *simulate* its view of an interaction with the prover, when given only the statement as input – i.e., without any knowledge of why the statement is true.

In many settings, however, the above security guarantee is not sufficient. Consider a situation in which Alice is giving a zero-knowledge proof of the statement X to Bob, and at the same time Bob is trying to give a zero-knowledge proof of some other statement X' to Charlie. Our intuitive definition of zero-knowledge tells us that Bob should not get any "help" in proving X' to Charlie by means of the zero-knowledge proof that Bob is getting from Alice – i.e. Bob should only be able to prove X' to Charlie if he could have done it on its own, without any help from Alice. This property is called *non-malleability* [14] for zero-knowledge proofs. It turns out that the standard simulation definition of zero knowledge does not imply non-malleability, and in fact, many known zero-knowledge proofs are susceptible

Proceedings of the 47th Annual IEEE Symposium
on Foundations of Computer Science (FOCS'06)
0-7695-2720-5/06 $20.00 © 2006 IEEE

to this kind of attack. We note that we can describe non-malleability as security in the following scenario: there are two executions of zero-knowledge proofs, with the adversary corrupting the verifier in one execution and the prover in the other.

Another setting considered in the literature is the following: Suppose there are many verifiers, all of which are receiving zero-knowledge proofs from various provers at the same time. We would like to guarantee that even if many of these verifiers collude, they still can't learn anything non-trivial from the provers – i.e., that it is possible to efficiently simulate the view of all the colluding verifiers interacting with the provers, given only the statements being proven by the provers. This property is called *concurrent zero knowledge* [15, 36], and here too, the standard definition of zero knowledge does not imply concurrent zero knowledge.

1.1. Our Results

In this work, we present the first protocol that is provably *simultaneously* non-malleable and concurrent zero knowledge in the "plain" cryptographic model without any setup assumptions. Our protocol allows provers to prove any **NP** statement and is based on standard cryptographic assumptions – namely, the existence of collision-resistant hash functions. The assumptions that we use is the existence of statistically hiding commitment schemes. Such schemes can be constructed with $O(n)$ rounds under one-way permutations [28] and and even regular (and in particular one-to-one) one-way functions [19] and in constant rounds under claw-free permutations [18] or collision-resistent hash functions [13, 20]. Simultaneous non-malleability and concurrency means that in the setting where there are many verifiers and provers all interacting concurrently, with scheduling decided by the adversary as well, security is preserved even if the adversary corrupts an arbitrary subset of *both* the provers and the verifiers. The definition of security is that for any such adversary there exists a polynomial-time simulator that, given only the statements proven by the honest parties (and not the witnesses), simulates the entire execution, and outputs along with the simulated transcript a list of witnesses corresponding to all statements successfully proven in this transcript by corrupted provers to honest verifiers. This definition is the natural combination of non-malleable zero knowledge [14] and concurrent zero knowledge [15, 36], and is also similar to the analogous definitions for non-malleable and concurrent commitments [14, 32]. We note that the best previous results on zero knowledge either (1) achieved only concurrent zero knowledge without non-malleability [36, 23, 34], (2) achieved non-malleability but only with a bounded number of parties present [14, 2, 33], (3) made use of global setup assumptions like a common reference strings [12] or time-delayed messages [21], or (4) used different security frameworks like super-polynomial simulation [35, 6, 27].

As in previous works on concurrent zero knowledge and non-malleable zero knowledge, our model assumes that the vector of inputs (statements and witnesses) to all parties is fixed according to some pre-determined distribution (although corrupted parties of course do not have to use their given inputs and can choose their inputs and messages adaptively). However, our security proof does *not* extend to the case of adaptively chosen honest inputs; this is with good reason, as it was shown by Lindell that there is *no* concurrent non-malleable zero knowledge protocol for honest adaptive inputs [26]. Indeed, Lindell's argument also ruled out many other functionalities, including oblivious transfer (OT), in the setting where the inputs for honest parties can be chosen adaptively based on outputs of previous protocols.

This leads to a natural question: Can we generalize our positive result on concurrent non-malleable zero knowledge to obtain a result for *any* polynomial-time functionality – as long as the inputs to honest parties are fixed in advance? We answer this question *negatively* by exhibiting a simple and natural functionality that is impossible to realize, even in the setting where all honest inputs are fixed in advance. Our negative result is also somewhat surprising since in many other settings (i.e., UC security in the common reference string model [12], bounded-concurrent security [24, 31, 30], super-polynomial simulation [35, 6, 27], and composition in timing model [21]) obtaining composable zero-knowledge protocols was the key step to obtaining protocols for all functionalities[1].

Our techniques. Perhaps surprisingly, our protocol does not use non-black-box techniques, but rather only uses black-box concurrent zero knowledge and non-malleable commitments; both tools that have been around for several years by now [36, 14] (although we do require some tweaking of these protocols, see below and Section 2). We see our main novelty in our proof of security.

Essentially all known techniques for achieving concurrent zero knowledge simulation and non-malleability in the plain model have relied crucially on proof techniques based on complex "rewinding" arguments[2]. A critical component to many results (e.g. [14, 34, 32, 6]) has been the development of new proof techniques to tame the complexity introduced by rewinding, often through new kinds of hybrid

[1] We do believe that the pattern will still hold true here – that our concurrent non-malleable zero-knowledge protocol will lead to protocols for all or large classes of functionalities, but just not according to the same definition of security. In the conclusions section, we mention some possible directions.

[2] We note that all known non-black-box techniques [1, 2, 29, 31, 33, 32, 6] for achieving concurrent simulation or non-malleability can also be seen as introducing complexities similar to those that arise with rewinding. This is one of the reasons that natural generalizations of [1] has not led to a constant-round concurrent zero-knowledge protocol.

Proceedings of the 47th Annual IEEE Symposium
on Foundations of Computer Science (FOCS'06)
0-7695-2720-5/06 $20.00 © 2006 IEEE

arguments. At a technical level, we continue in this line and develop new techniques for dealing with complex rewinding in security proofs.

Our protocol uses the Prabhakaran-Rosen-Sahai (PRS) [34] concurrent zero knowledge protocol and simulation strategy. We also want to make use of non-malleable commitment constructions (e.g. [14, 32]) to obtain non-malleability. This gives rise to two main obstacles: (1) We need to guarantee that the non-malleability properties of these commitment schemes remain even in the presence of our rewinding. Note that in general, this should not be true – an adversary for a plain-model non-malleable commitment scheme such as [14, 32] that can rewind honest parties would always be able to cheat. We develop a new hybrid argument that shows that we can guarantee non-malleability by making specific use of the properties of the PRS rewinding strategy and a statistical zero knowledge variant of the PRS protocol. (2) The other major obstacle is that the techniques for non-malleability necessarily involve rewinding of their own (for extraction). We develop a new proof technique to show that the extraction methods we need can work "on top of" the PRS rewinding strategy.

For our impossibility result ruling out concurrent non-malleable realizations of more general functions, even when honest party input distributions are fixed, we work as follows: we start by taking one of the counterexamples showing that very strongly composable protocols (e.g., UC security [8] or security against "chosen-protocol attack" [22, 25]) for, say, zero knowledge, do not exist in the plain model (where there are no trusted parties or common reference strings). This basically implies that for every supposedly composable zero-knowledge argument Π, there exists a protocol Π', depending on Π, such that their concurrent execution is not secure. The main novelty in our work is that in order to get the kind of result we want, we use a variant of Yao's garbled circuit technique [38] to "compile" the protocol Π' into a protocol using the oblivious transfer functionality. Thus, we create a scenario where for every protocol Π implementing the combined zero knowledge and oblivious transfer functionality (or equivalently, for every pair of protocols Π_{ZK} and Π_{OT} each implementing these two functionalities), there's an adversary launching a concurrent attack that manages to learn a secret with probability close to 1 in the real world, but no adversary would only be able to learn the secret with non-negligible probability in the ideal model. Note that, unlike its typical use, we're using Yao's technique here to get a *negative* result. (This is somewhat similar in spirit to [4].)

1.2. Previous works.

Concurrent zero knowledge. Concurrent zero knowledge (where the adversary corrupts either only provers or only verifiers) was defined by Dwork, Naor and Sahai [15] and the first construction in the standard model was given by Richardson and Kilian [36]. The number of rounds was improved to $\tilde{O}(\log n)$ by [23, 34] which is optimal for *black-box simulation* [10]. (A constant round protocol for bounded-concurrent zero knowledge was given in [1] using non-black-box simulation.) **Non-malleable zero knowledge.** Non-malleable zero knowledge was first defined and constructed by Dolev, Dwork and Naor [14]. Constant round protocols were given in [2, 33]. These latter works also introduced some more convenient definitions (which we follow) than the [14] definition (inspired by definitions of non-malleable *non-interactive* zero knowledge [37]). **Non-malleable and concurrent commitments.** By a simple hybrid argument, every commitment scheme remains secure under concurrent composition if the adversary can corrupt either only senders or only receivers. As in the case of zero knowledge, stand-alone non-malleable commitments were defined by [14] and constant-round protocols were given in [2, 33]. Pass and Rosen [32] showed that the commitment scheme from [33] is actually *concurrently non-malleable* thus giving an $O(1)$ round concurrent non-malleable commitment scheme. **Note:** In many previous works, progress in commitment schemes and zero-knowledge went hand in hand, where one could obtain a ZK protocol satisfying security notion X by plugging a commitment scheme satisfying X to a standard standalone protocol [14, 9, 12, 24]. Thus, one might hope that one could obtain in this way a concurrent non-malleable ZK protocol from the [32] scheme. However, an important limitation of [32] is that security is guaranteed only under the condition that only the *commit* protocol and not the *reveal* protocol is executed concurrently. For this reason, such commitment schemes do not automatically imply concurrent non-malleable zero knowledge proofs. In particular, we do not know that if we plug in [32]'s commitments in one of the well known constant-round ZK or honest-verifier ZK protocols we will get a concurrent non-malleable ZK protocol. In fact, that would be quite surprising since in particular it will yield the first constant round *concurrent zero knowledge* protocol. We note that our work here does not work in this way, and indeed, we can make use of "*non-concurrent*" non-malleable commitment protocols like the original protocol of [14], thus avoiding non-black-box techniques altogether, and reducing our assumptions to just regular one-way functions. We also don't know whether it's possible make the proof simpler by using concurrently non-malleable commitments.

Universally composable (UC) security, general and self composition. In [8], Canetti introduced the notion of *universally composable* or UC security for cryptographic protocols. This is a very strong notion of security and in particular a UC secure zero-knowledge protocol will be concurrently non-malleable and in fact will compose with an

Proceedings of the 47th Annual IEEE Symposium
on Foundations of Computer Science (FOCS'06)
0-7695-2720-5/06 $20.00 © 2006 IEEE

environment that contains executions of arbitrary other protocols as well (see also [25]). However, this notion, that essentially implies black-box straightline simulation, is in some sense "too strong", and it was shown that in the "plain" model, without trusted parties, honest majority or setup, it is *impossible* to achieve UC-secure zero knowledge and in fact a very wide range of functionalities including commitment schemes [8, 9, 11]. (See [7, 8, 12, 3] for constructions in other models.) **Self-composition.** As mentioned above, Lindell [26] showed that for the case of *message passing functionalities* (functionalities allowing to transmit a bit, in particular including zero knowledge), security for concurrent composition of the *same* protocol *under adaptive input selection* essentially implies UC security and hence it is impossible to obtain a zero knowledge protocol satisfying this notion of self-composition in the plain model. Adaptive input selection is defined by having the inputs supplied by an environment as in the UC model, but unlike the UC definition, this environment is not allowed to look at the actual *communication* of the executions but only at the *outputs* of these executions. In contrast, in our security model the inputs may be chosen from some distribution but are supplied in advance to all parties, and so, while we can't control the corrupted parties' behavior, the honest parties do not choose their inputs adaptively based on previous executions.

Super-polynomial-time simulation. Another sequence of works considered a setting where the ideal model simulator is allowed to run in *super-polynomial* time [35, 6, 27]. This allows to bypass the UC impossibility results and yield protocols for any functionality that seem to supply adequate security for many applications. However, the definition is not as intuitive and mathematically clean as polynomial-time simulation, and the current constructions do suffer from drawbacks such as requiring stronger complexity assumptions, and a tradeoff between the time of simulation and the standalone soundness of the protocol. **Security for independent inputs.** Garay and MacKenzie [16] show a protocol for oblivious transfer that is concurrently secure if the inputs to the parties in each execution is chosen independently and at random from a known distribution such as the uniform distribution. We note that in this paper we consider the more standard setting where the inputs are arbitrarily chosen and in particular may be correlated.

1.3. Preliminaries.

We consider only two party protocols in this paper. Our model is of m parties $P_1, \ldots, P_m$ (not necessarily aware of one another) that interact in pairs via some two party protocol Π. There's some distribution D on inputs $x_1, \ldots, x_m$ and each party P_i uses input x_i in its interaction (by adding more parties if necessary, we can assume that each party participates in at most one interaction of Π). We assume an adversary $\mathtt{Adv}$ that chooses initially to corrupt a set of parties $\{P_i : i \in C\}$, and receives the inputs for that set, and completely controls these parties. The adversary can also schedule concurrently and adaptively all the messages in the network. We assume that all parties in the network have unique identities and authenticated communication (following [14] this can be relaxed somewhat for the positive result). We say that Π *securely implements* an ideal functionality $\mathcal{F}$ with two inputs and two outputs if for any such $\mathtt{Adv}$ corrupting a set C there's a simulator Sim that receives the inputs x_i for $i \in C$, and for every pair (i, j) that interacts via Π with $i \in C$ and $j \notin C$, Sim gets one access to the first output of the function $x \mapsto \mathcal{F}(x, x_j)$ (we have an analogous definition if the corrupted party is the second in the pair). The outputs of Sim and the second output should be computationally indistinguishable from the outputs of $\mathtt{Adv}$ and the outputs of the honest parties in the real execution. It can be shown that Π is concurrent non-malleable zero knowledge for an $\mathbf{NP}$-relation R if and only if it secure implements the ZKPOK functionality $\mathcal{F}$ defined as follows: $\mathcal{F}(x \circ w) = x$ iff $(x, w) \in R$ and $\mathcal{F}(x \circ w) = \perp$ otherwise (this functionality only uses one of its inputs).

2. A concurrent non-malleable zero knowledge protocol

Definition. The formal definition of a Concurrent Non-Malleable Zero Knowledge (CNMZK) argument of knowledge for membership in an $\mathbf{NP}$ language is provided in the full version [5]. Informally, it is an interactive proof protocol with the usual completeness property and a *concurrent non-malleable* security property. The latter states that for every (non-uniform PPT) adversary $\mathcal{A}$ interacting with honest provers $P_1, \ldots, P_{m_L}$ in m_L "left sessions" and with honest verifiers $V_1, \ldots, V_{m_R}$ in m_R "right sessions" of the protocol (with $\mathcal{A}$ controlling the scheduling of all the sessions), there exists a (non-uniform PPT) simulator $\mathcal{S}$ such that for every set of "left inputs" $y_1, \ldots, y_{m_L}$, we have $\mathcal{S}(y_1, \ldots, y_{m_L}) = (\nu, z_1, \ldots, z_{m_R})$, where ν is a simulated view of $\mathcal{A}$,[3] and $z_1, \ldots, z_{m_R}$ are valid witnesses to the statements proven by $\mathcal{A}$ to $V_1, \ldots, V_{m_R}$ according to the view ν. (We let $z_i = \perp$ if V_i does not accept according to ν.)

Note that the above security property subsumes both zero-knowledge and proof-of-knowledge properties. This definition is in the same spirit as security definitions using ideal functionalities (as, for instance, in the UC model [8]): the witnesses that $\mathcal{S}$ produces (i.e., extracts from $\mathcal{A}$) can be considered what it sends to the ideal functionality. We remark that security only requires that the extracted witnesses

[3] Here, and elsewhere, by the view of a party we mean the sequence of its internal states during the execution, including the messages received and sent by it.

Proceedings of the 47th Annual IEEE Symposium
on Foundations of Computer Science (FOCS'06)
0-7695-2720-5/06 $20.00 © 2006 IEEE

be valid, and makes no reference to what witnesses $\mathcal{A}$ "actually uses."

Result from [34]. We heavily rely on techniques from [34]. First we sketch the "protocol preamble" used there.

1. **PRS Commitment:** The verifier picks a (sufficiently long) random string σ, commits to σ and many secret sharings of σ, using a statistically binding commitment scheme $\mathsf{Com}_{\mathsf{PRS}}$.
2. **PRS Challenge-Response:** This is followed by (super-logarithmically) many rounds of random challenges by the prover. In response, the verifier must open some of the PRS commitments (without revealing σ).
3. The prover considers the preamble to have "concluded."
4. **PRS Opening:** The verifier opens all the commitments made in the PRS Commitment step, and the prover verifies consistency.
5. The prover "accepts" the preamble.

There can be other messages in the protocol between the prover concluding the preamble and the verifier opening the commitments.

The *PRS simulator* (for our purposes) is the following program which "simulates" multiple (polynomially many in the security parameter) concurrent sessions of the protocol between honest provers and a combined adversarial verifier, $\mathcal{A}_{\mathsf{PRS}}$. The simulator gets inputs of all the parties in all the sessions, and it runs the honest provers and the adversarial verifier internally.[4] In the end it produces an ordered list of "threads of execution." A thread of execution consists of views[5] of all the parties, such that the following hold.

- Each thread of execution is a perfect simulation of a prefix of an actual execution.
- The last thread, called the *main thread*, is a perfect simulation of a complete execution (i.e., until all the parties terminate); all other threads are called *look-ahead threads*.
- Each thread shares a (possibly empty) prefix with the previous thread, and is derived by running the honest parties with fresh randomness after that point.

The aim of the PRS simulator is, for each PRS commitment that it comes across in any session in any thread, to extract the committed value σ (referred to as the "PRS secret") before the preamble is *concluded* in that thread. The extraction is achieved by observing the adversary's messages in multiple previous threads. If it fails to extract the PRS secret in any session in a thread, *and* the execution goes on to *accept* the preamble of that session in that thread, then the simulation is said to "get stuck." [34] guarantees that the probability of the PRS simulation getting stuck is negligible.

Lemma 2.1. *(Adapted from [34]) Consider provers $P_1, \ldots, P_m$ and an adversarial verifier $\mathcal{A}_{PRS}$ running m sessions of a protocol with the PRS preamble as described above, where m is any polynomial in the security parameter k. Then except with negligible probability, in every thread of execution output by the PRS simulator, if the simulation reaches a point where the prover P_i accepts the PRS preamble with σ as the secret in the preamble, then at the point when the preamble was concluded, the simulator would have already recorded the value σ.*

In fact [34] prove a refinement of this lemma (that we too will need): instead of the simulator running each thread exactly as in the original execution, if each thread (individually) is executed in an indistinguishable way, the lemma still holds. It is important that here we require the indistinguishability requirement only on a *per thread* basis. In particular the joint distribution of the threads in the latter simulation is allowed to be distinguishable from the joint distribution of the threads in the original simulation.

We shall adapt the PRS simulator to our setting in which an adversary $\mathcal{A}$ is engaged in concurrent left hand side sessions as the verifier, while concurrently playing the prover in multiple right hand sessions. In (unshared parts of) the different threads, the simulator uses fresh randomness for all the honest parties, but uses the same random tape for $\mathcal{A}$ in all the threads. This is important for us because in our simulation we will need to use fresh randomness for the right hand side verifiers in different threads (except during the shared prefixes).

Non-Malleable Commitment. Another ingredient we need is a perfect (or statistically) binding, non-malleable (not necessarily concurrent non-malleable) commitment with a "stand-alone extractability" property. The non-malleability property is similar to that defined in [32], but needs to hold when there is one left and right executions each. The construction in [14] also satisfies this property. The "extractability" property is that there is an efficient extractor which, given a randomly generated view of a stand-alone committer committing a value to an honest receiver, can extract the committed value except with negligible probability. We also impose a technical condition that the receiver should be public coin up to a "knowledge determining message" in the protocol. Protocols in [14] and [32] can be easily modified to have these properties, as shown in the full version [5].

[4] Note that the "simulator" as described here is given all the inputs to all the parties. Later, after introducing this simulator into the sequence of hybrids in our proof, we shall show how to get rid of these inputs.

[5] Here, and elsewhere, by the view of a party we mean the sequence of its internal states during the execution, including the messages received and sent by it.

Proceedings of the 47th Annual IEEE Symposium
on Foundations of Computer Science (FOCS'06)
0-7695-2720-5/06 $20.00 © 2006 IEEE

Other ingredients. The other ingredients we use are a statistically (or perfectly) hiding commitment scheme $\mathsf{Com}_{\mathsf{SH}}$ and a statistical (or perfect) ZK argument of knowledge sZKAOK, for proving knowledge of witness for membership in any **NP** language.

We note that all our ingredients are realizable under the assumption that regular one-way functions exist [28, 19].

2.1. Our Protocol

Consider an **NP**-complete language L with a witness relationship R. The prover and verifier receive a common input y and the prover receives a witness w such that $R(y, w) = 1$. The protocol CNMZK is described below.

Phase I: PRS preamble up to the point where the prover *concludes* the preamble.

Phase II: Prover commits to the all-zero string using $\mathsf{Com}_{\mathsf{SH}}$. Then it uses sZKAOK to prove the knowledge of the randomness and inputs to this execution of $\mathsf{Com}_{\mathsf{SH}}$.

Phase III: Continue the PRS preamble until the prover accepts the preamble. Let the secret in the preamble (as revealed by the verifier) be σ.

Phase IV: Prover commits to the witness w using $\mathsf{Com}_{\mathsf{NM}}$.

Phase V: Prover proves the following statement using sZKAOK: either the value committed to in Phase IV is w such that $R(y, w) = 1$, or the value committed to in Phase II is σ. It uses the witness corresponding to the first part of the statement.

Theorem 2.2. *Protocol* CNMZK *is a black-box concurrent non-malleable zero knowledge argument for membership in the* **NP** *language* L.

Proof Sketch: It is easy to see that the protocol satisifies the completeness condition. Below we sketch how to build a simulator-extractor, as required by the definition.

We build the simulator $\mathcal{S}$ in stages, via intermediate simulators $\mathcal{H}_i$, for $i = 1, \ldots, 4$. $\mathcal{H}_i$ outputs a simulated view $\nu^{(i)}$. ($\mathcal{S}$ will in addition output a list of witnesses.) We define $2m_R$ random variables $\{b_\ell^{(i)}, \alpha_\ell^{(i)}\}_{\ell=1}^{m_R}$, where $b_\ell^{(i)}$ is a bit denoting whether according to $\nu^{(i)}$, V_ℓ accepted the proof from the adversary or not, and $\alpha_\ell^{(i)}$ is the value contained in the Phase IV commitment $\mathsf{Com}_{\mathsf{NM}}$ received by V_ℓ (as determined by the determining message; if there is no unique value, then it is defined to be $\perp$).

Stage 1: $\mathcal{H}_1$ gets all the inputs to $P_1, \ldots, P_{m_L}$ as well as the inputs to $\mathcal{A}$. It internally runs the (honest) programs of $P_1, \ldots, P_{m_L}$, as well the honest program for the verifiers $V_1, \ldots, V_{m_R}$, to generate $\mathcal{A}$'s view $\nu^{(1)}$. The simulation is perfect.

Also one can show that due to the knowledge soundness of the sZKAOK scheme used in Phase II and Phase V, if V_ℓ accepts the proof in the ℓ-th right hand session in the simulated view ν, then, except with negligible probability, the Phase IV commitment in that session indeed contains a valid witness z_ℓ to the statement x_ℓ. That is, except with negligible probability,

$$\forall \ell \qquad \left(b_\ell^{(1)} = 1\right) \implies \left(R(x_\ell, \alpha_\ell^{(1)}) = 1\right). \qquad (1)$$

Stage 2: $\mathcal{H}_2$ works just like $\mathcal{H}_1$, but it also does the PRS look-aheads and records the PRS secrets. Recall that this means that the simulator runs many perfect simulations of the execution with shared prefixes (but using fresh randomness in the unshared parts), and records the PRS secrets for each preamble concluded in any thread. $\mathcal{H}_2$ aborts if the PRS simulation gets stuck. Otherwise it outputs the view of the adversary in the main thread of this simulation as $\nu^{(2)}$. By Lemma 2.1 we know that the probability of aborting is negligible. Hence, we have $\nu^{(1)} \equiv_{\mathrm{s}} \nu^{(2)}$ and $\forall \ell\ (b_\ell^{(1)}, \alpha_\ell^{(1)}, y_\ell) \equiv_{\mathrm{s}} (b_\ell^{(2)}, \alpha_\ell^{(2)}, y_\ell)$.

Stage 3: $\mathcal{H}_3$ works like $\mathcal{H}_2$, except that in all the simulated left hand side sessions, the prover commits to *the PRS secrets* in the Phase II $\mathsf{Com}_{\mathsf{SH}}$, and follows up with an honest execution of sZKAOK for this commitment. Since $\mathsf{Com}_{\mathsf{SH}}$ is a statistically hiding commitment scheme, and sZKAOK is statistical zero knowledge we get $\nu^{(2)} \equiv_{\mathrm{s}} \nu^{(3)}$ and $\forall \ell\ (b_\ell^{(2)}, \alpha_\ell^{(2)}, y_\ell) \equiv_{\mathrm{s}} (b_\ell^{(3)}, \alpha_\ell^{(3)}, y_\ell)$.

Stage 4: The heart of the proof is in building $\mathcal{H}_4$, which does not need the left provers' inputs w_j any more. It works like $\mathcal{H}_3$, except that in all the simulated left hand side sessions, the prover commits to the all zeros string in the Phase IV $\mathsf{Com}_{\mathsf{NM}}$, and uses the $\mathsf{Com}_{\mathsf{SH}}$ commitment as the witness in the Phase V sZKAOK instead of the witnesses w_j. We delay the main part of the proof, which requires the non-malleability property of the commitment scheme $\mathsf{Com}_{\mathsf{NM}}$, and instead state the following claim first.

Claim 2.3. $\nu^{(3)} \equiv_c \nu^{(4)}$ *and* $\forall \ell\ (b_\ell^{(3)}, \alpha_\ell^{(3)}, y_\ell) \equiv_c (b_\ell^{(4)}, \alpha_\ell^{(4)}, y_\ell)$.

Stage 5: Finally we describe the simulator-extractor $\mathcal{S}$. First it runs $\mathcal{H}_4$ to produce a view of the adversary, $\nu^{(4)}$. Then it extracts the values $\alpha_\ell^{(4)}$, for $\ell = 1, \ldots, m_R$. For extracting thus, for each ℓ, $\mathcal{S}$ will consider $\mathcal{H}_4$ as a standalone adversary $\mathcal{A}_\ell^*$ making a single commitment to an external receiver, and then invokes the extractor with (appropriately reformatted) view $\nu^{(4)}$ and $\mathcal{A}_\ell^*$ as the committer which produced this view.

Unfortunately this is complicated by the fact that in the PRS simulation, $\mathcal{H}_4$ needs to run look-ahead threads and rewind before it can run the main thread. Thus a straightforward construction of $\mathcal{A}_\ell^*$ will require it to be able to rewind the external receiver. Nevertheless, using the condition that the receiver in the $\mathsf{Com}_{\mathsf{NM}}$ protocol uses no private coins till the knowledge determining message, we show

Proceedings of the 47th Annual IEEE Symposium
on Foundations of Computer Science (FOCS'06)
0-7695-2720-5/06 $20.00 © 2006 IEEE

how the PRS simulation can be continued without having to rewind the external receiver.

The final output of $\mathcal{S}$ is $(\nu, \beta_1, \ldots, \beta_{m_R})$ where β_ℓ are the values extracted as described above. By the extraction guarantee, if according to ν, V_ℓ accepted the proof, and in particular accepted the Phase IV commitment, then $\beta_\ell = \alpha_\ell^{(4)}$ except with negligible probability.

From the above, we get $\nu^{(4)} \equiv_c \nu^{(1)}$, where the former is the view generated by $\mathcal{S}$ and the latter is identical to that of the adversary $\mathcal{A}$ in an actual execution. Further, we have $\forall \ell \ \left(b_\ell^{(4)} = 1\right) \implies \left(R(x_\ell, \alpha_\ell^{(4)}) = 1\right)$ except with negligible probability. This follows from Equation 1, the fact that $(b_\ell^{(4)}, \alpha_\ell^{(4)}) \equiv_c (b_\ell^{(1)}, \alpha_\ell^{(1)})$ as implied by the above, and the fact that the condition $\left(b_\ell^{(\cdot)} = 1\right) \implies \left(R(x_\ell, \alpha_\ell^{(\cdot)}) = 1\right)$ can be efficiently checked.

This completes the proof except for the proof of Claim 2.3. □

2.1.1 Proof of Claim 2.3

This is the most delicate part of the proof, which reduces the concurrent non-malleability of our zero-knowledge protocol to (non-concurrent) non-malleability of the commitment scheme $\mathsf{Com_{NM}}$. The goal is to show that in moving from the hybrid $\mathcal{H}_3$, which uses the real left hand side witnesses in the simulation, to $\mathcal{H}_4$ which uses the alternate PRS witnesses and commits to all-zeros strings instead of the witnesses, the values committed to by the adversary do not change adversely. Conceptually the difficulty is in separating the effect of the modifications in the left sessions from those in the right sessions. The technical difficulties stem from the somewhat intricate nature of PRS simulation which causes change at some point in the simulation to propagate in subtle ways.

Before proceeding we point out, intuitively, why we *do not* require *concurrent* non-malleability for $\mathsf{Com_{NM}}$: all we require is that, in $\mathcal{H}_4$, for each right hand session, the commitment made using $\mathsf{Com_{NM}}$ continues to be a witness, if it used to be a witness in $\mathcal{H}_3$; we *do not* require that the entire set of committed values remain indistinguishable jointly.

We move from $\mathcal{H}_3$ to $\mathcal{H}_4$ using a carefully designed series of hybrid simulators $\tilde{\mathcal{H}}_{i:1}$ and $\tilde{\mathcal{H}}_{i:2}$. To describe these hybrids, first we introduce some notation. In the PRS simulation consider numbering (in order) all the occurrences of the first message (FM) in the Phase IV $\mathsf{Com_{NM}}$ in the left hand side sessions. Note that in a full PRS execution, due to the look-aheads, we may have multiple FMs being sent by the same left hand side prover (though only one in each thread). Further, in the simulation, for any i, the left hand prover sending FM_i is a random variable with support on all m_L provers: this is because in each thread, the adversary *dynamically schedules* the protocol sessions based on the history of messages in the thread (and its random tape, which we have fixed). We shall denote the index of the left hand prover sending FM_i by $p(i)$. We will refer to the instances of sZKAOK provided by $P_{p(i)}$ in threads passing through FM_i, as "belonging" to FM_i.

We define $\tilde{\mathcal{H}}_{0:2}$ to be $\mathcal{H}_3$ and let $\mathcal{H}_4$ be $\tilde{\mathcal{H}}_{N:2}$, where N is an upperbound on the number of FMs in the PRS schedule. For $i = 1, \ldots, N$, the simulators $\tilde{\mathcal{H}}_{i:1}$ and $\tilde{\mathcal{H}}_{i:2}$ are as follows:

$\tilde{\mathcal{H}}_{i:1}$: Exactly like $\tilde{\mathcal{H}}_{i-1:2}$, except that for all the sZKAOK belonging to FM_i, the prover will use the corresponding PRS secret as the witness (instead of using $w_{p(i)}$). If the PRS secret is not available, then the simulator fails[6].

$\tilde{\mathcal{H}}_{i:2}$: Exactly like $\tilde{\mathcal{H}}_{i:1}$, except that in FM_i the prover commits to the all-zeros string (instead of $w_{p(i)}$) and continues the execution accordingly.

For $i = 1, \ldots, N$ we define random variables $\tilde{\nu}^{(i:1)}$ and $\{\tilde{b}_\ell^{(i:1)}, \tilde{\alpha}_\ell^{(i:1)}\}_{\ell=1}^{m_R}$ and $\tilde{\nu}^{(i:2)}$ and $\{\tilde{b}_\ell^{(i:2)}, \tilde{\alpha}_\ell^{(i:2)}\}_{\ell=1}^{m_R}$ analogous to $\nu^{(1)}$ and $\{b_\ell^{(1)}, \alpha_\ell^{(1)}\}_{\ell=1}^{m_R}$. Note that we need to show that $\tilde{\nu}^{(0:2)} \equiv_c \tilde{\nu}^{(N:2)}$ and $\forall \ell \ (\tilde{b}_\ell^{(0:2)}, \tilde{\alpha}_\ell^{(0:2)}, y_\ell) \equiv_c (\tilde{b}_\ell^{(N:2)}, \tilde{\alpha}_\ell^{(N:2)}, y_\ell)$. We do this via the following sequence:

$$\tilde{\nu}^{(i-1:2)} \equiv_c \tilde{\nu}^{(i:1)} \tag{2}$$

$$\tilde{\nu}^{(i:1)} \equiv_c \tilde{\nu}^{(i:2)} \tag{3}$$

$$\forall \ell \qquad (\tilde{b}_\ell^{(i-1:2)}, \tilde{\alpha}_\ell^{(i-1:2)}, y_\ell) \equiv_c (\tilde{b}_\ell^{(i:1)}, \tilde{\alpha}_\ell^{(i:1)}, y_\ell) \tag{4}$$

$$\forall \ell \qquad (\tilde{b}_\ell^{(i:1)}, \tilde{\alpha}_\ell^{(i:1)}, y_\ell) \equiv_c (\tilde{b}_\ell^{(i:2)}, \tilde{\alpha}_\ell^{(i:2)}, y_\ell) \tag{5}$$

It is not hard to argue that going from $\tilde{\mathcal{H}}_{i-1:2}$ to $\tilde{\mathcal{H}}_{i:1}$, the main thread remains statistically indistinguishable. One subtlety here is that though the Phase V sZKAOK remains statistically indistinguishable when the alternate witness is used, indistinguishability does not hold when multiple threads are considered together. But the only way a thread can affect subsequent threads is through the availability of the PRS secrets at the right points in the simulation. Then, by the refinement mentioned after Lemma 2.1, it will hold that the PRS secrets will continue to be available as required except with negligible probability. Thus each individual thread, and in particular the main thread, continues to be statistically indistinguishable between the simulations by $\tilde{\mathcal{H}}_{i-1:2}$ and $\tilde{\mathcal{H}}_{i:1}$. This in turn implies both equations (2) and (4).

Equation (3) follows from the hiding property of $\mathsf{Com_{NM}}$. However to prove equation (5), this is not enough, because only the right hand side commitments appear in the simulated view and not the committed values themselves (which can be distinguishable even when the commitments themselves are indistinguishable). So now we build a machine M_ℓ which will "expose" the incoming left hand side

[6] as it would have already failed in $\mathcal{H}_3$

Proceedings of the 47th Annual IEEE Symposium
on Foundations of Computer Science (FOCS'06)
0-7695-2720-5/06 $20.00 © 2006 IEEE

commitment from $P_{p(i)}$ and the outgoing right hand side commitment to V_ℓ. That is, M_ℓ will interact with an external sender and an external receiver for these commitments, while internally simulating the rest. Then we shall use the non-malleability property of $\mathsf{Com}_{\mathsf{NM}}$ to argue that the values committed to by M_ℓ in two experiments – one in which $P_{p(i)}$ commits to $w_{p(i)}$ and another in which to the all-zeros string – are indistinguishable, and hence so will be the values committed to V_ℓ by $\tilde{\mathcal{H}}_{i:1}$ and $\tilde{\mathcal{H}}_{i:2}$.

But the precise argument is more involved, because we need to take into account whether the right hand commitment to V_ℓ occurs before, after or overlapping with FM_i (which is the first message of $P_{p(i)}$). The most interesting case is when FM_i occurs in the main thread, before the first message (or more precisely the determining message) of the commitment to V_ℓ is sent. The key step in building M_ℓ is being able to run the main thread of the PRS simulation in $\tilde{\mathcal{H}}_{i:1}$ and $\tilde{\mathcal{H}}_{i:2}$ without having to rewind the external receiver or the committer. We show that given the way we have defined the ordering on the FMs and the hybrids $\tilde{\mathcal{H}}_{i:1}$ and $\tilde{\mathcal{H}}_{i:2}$, M_ℓ can run the part of the main thread after FM_i without running any further look-ahead threads.

Once we build M_ℓ it is routine to show that the non-malleability condition on $\mathsf{Com}_{\mathsf{NM}}$ implies equation (5). □

3. Impossibility result for concurrent non-malleable general functionalities.

In this section we sketch our negative result, showing that it is *impossible* to extend our result for zero knowledge to every functionality. We will only sketch the proofs here, and refer the reader to the full version [5] for further details.

We need to show that there is some polynomial-time function $\mathcal{F}$, such that for every protocol implementing $\mathcal{F}$, there's a concurrent attack that can be carried in the real model and cannot be carried in the ideal mode, even in the case where all honest parties' inputs are chosen according to some (correlated) distribution and fixed in advance. Our function $\mathcal{F}$ will take two inputs and have one output. We call the party supplying the first input the *sender* and the party supplying the second input the *receiver*. By our convention only the receiver gets the output of the combination. We will define $\mathcal{F}$ to be a combination of the zero knowledge and the oblivious transfer (OT) functionalities (an equivalent way to state our results is that there are no pairs of protocols for zero knowledge and OT that compose with another). More formally, let $f : \{0,1\}^k \to \{0,1\}^k$ be a one-way function (where k is some security parameter) and R_f be the $\mathbf{NP}$-relation $\{(x,w) : w = f(x)\}$. We let $\mathcal{F}_{ZK}(x \circ w, x) = 1$ if $(x,w) \in R$ and zero otherwise. We let $\mathcal{F}_{OT}(x_1 \circ x_2, b) = x_b$ where $x_1, x_2 \in \{0,1\}^k$ (that is, we use the variant of OT known as $\binom{2}{1}$ string OT). The functionality $\mathcal{F}$ will simply be a combination of $\mathcal{F}_{ZK}$ and $\mathcal{F}_{OT}$. That is, we will have an for $\mathcal{F}$ additional input bit specified by each party, and if both parties use zero for this bit $\mathcal{F}$ will apply $\mathcal{F}_{ZK}$ on the rest of the inputs, if both use one $\mathcal{F}$ will apply $\mathcal{F}_{OT}$, and otherwise (if they don't agree on this bit) $\mathcal{F}$ will output $\perp$.

Our main theorem of this section is the following

Theorem 3.1. *Assume that f is a one-way function and let $\mathcal{F}$ be defined as above.Let Π be any polynomial-time two party protocol that computes $\mathcal{F}$ if both parties are honest. Then, there's a polynomial $t(\cdot)$ such that for any k there exists distribution D on $2t = t(k)$ inputs for Π, a concurrent scheduling S of t executions of Π, a polynomial-time adversary A, and a polynomial function* SECRET *that maps the inputs into $\{0,1\}^k$.*

- *In a concurrent execution of t copies of Π according to the schedule S with the honest parties and the corrupted parties receiving inputs chosen from the distribution D, the adversary A outputs the value of* SECRET *on the inputs with probability* 1.
- *In an ideal model, for any polynomial-time adversary $\hat{A}$ that gets access to the t copies of the ideal OT functionality, with the honest parties' inputs in these copies coming from D, (and $\hat{A}$ receiving the inputs corresponding to the corrupted parties) the probability that $\hat{A}$ outputs the value of* SECRET *on the inputs is negligible, where this probability is taken over D and the coins of $\hat{A}$.*

We note that since standalone OT implies the existence of one-way functions, and $\mathcal{F}$ subsumes OT, this theorem implies unconditionally that no protocol can realize the functionality $\mathcal{F}$ and be self-composable, even when honest-party inputs are from a fixed distribution.

This is the first result ruling out composable protocols in the plain model for general (possibly non-black-box) simulation, honest inputs fixed in advance, and without requiring composability also with other arbitrary protocols. It's somewhat surprising since in many previous settings, (UC-security [12], bounded composition [24, 30], timing [21], super-polynomial simulation [35, 6]) obtaining a composable zero knowledge protocol implied obtaining a composable protocol for general functionalities.

3.1. Proof sketch of Theorem 3.1

The proof of Theorem 3.1 proceeds in two stages:

First stage. First, (as warm-up) we prove that for every protocol Π_{ZK} for the zero knowledge functionality (for the relation R_f above), there exists an ideal two-party deterministic function F_Π (that depends on the protocol Π_{ZK}) such that a single instance of Π_{ZK} executed concurrently with several ideal calls to copies of F_Π will not be secure. We start by considering the following scenario:

Proceedings of the 47th Annual IEEE Symposium
on Foundations of Computer Science (FOCS'06)
0-7695-2720-5/06 $20.00 © 2006 IEEE

1 Alice and David are honest, Bob and Charlie are malicious and coordinate. A value x is public and Alice and David share w such that $x = f(w)$.
2 Alice proves to Bob using Π_{ZK} that she knows w.
3 Charlie and David interact using the following protocol P: the protocol P tells David that if Charlie manages to run protocol Π_{ZK} as the prover showing knowledge of w, then David should send w to Charlie.[7]
4 Clearly, if Π_{ZK} and P are executed concurrently in this scenario than the malicious Bob and Charlie can learn w, even though they would not have been able to learn w if Π_{ZK} was replaced with an ideal call to the $\mathcal{F}_{ZK}$ functionality.

Our main tool in transforming P into a non-interactive functionality F_Π is to use Message Authentication Codes (MACs) to force the adversary to make calls to F_Π in a certain order, imitating an interactive protocol. Thus, instead of having one execution of the protocol P, we will have ℓ executions of a non-reactive function F_Π (where ℓ is the number of prover messages in Π_{ZK}). The sender of F_Π will have as input a secret MAC key, randomness for the protocol P above, and the secret input w. If the receiver's input is a partial transcript p of P (with the last message being David's) with a valid tag on p, and an additional message m of Charlie's, then F_Π will compute David's next message m' on the transcript $p \circ m$, and will output the transcript $p \circ m \circ m'$ and a tag on this message. One can see that getting access to ideal calls for $\mathcal{F}$ is not more (and not less) helpful than interacting with P.

Second stage. The reason we're not finished is not just because F_Π is a "less natural" functionality than $\mathcal{F}$, but also – and more importantly – because the function F_Π can (and will) depend on Π_{ZK} in its definition, its complexity and its input size. To get the negative result that we want, we need to go further and exhibit a functionality $\mathcal{F}$ that cannot be implemented by any Π.

The second conceptual stage is to take this scenario of the protocol Π_{ZK} and functionality F_Π and compile this into a scenario where the only thing executed in the network is one copy of a zero knowledge protocol and many copies of an OT protocol, with the honest parties' inputs for these copies chosen from a set of predefined distributions. We then argue that the previous real-world attack remains viable in this scenario and (more subtly) that it is still infeasible to perform this attack if all these copies were replaced by ideal calls to the OT/ZK functionalities. Since $\mathcal{F}$ is a combination of these functionalities, the result follows.

For this stage we will use a variant of Yao's garbled circuit technique [38]. Note that unlike its typical usage, we use here this technique to get a *negative* result (this is somewhat similar to what was done in [4]'s negative results for software obfuscation).

The overall idea is as follows: We will set up a situation – in *both* the ideal and real worlds – which could potentially allow for the evaluation of any function, using a variant of the garbled circuit technique and ideal calls to an OT functionality. But, we will set up the honest party inputs in such a way that the only functions that can be evaluated mimic the functionality $\mathbb{F}_\Pi$ described above. So here, the only functionalities are the ZK and OT functionalities, but the predetermined honest party inputs depend on the specific protocol Π_{ZK}. Then, in the real world, the adversary will always be able to win, whereas in the ideal world (where Π_{ZK} is not being executed), the adversary cannot win. The garbled circuits will not be sent out by any party (as we're not allowed to do anything on the network except run the protocol for $\mathcal{F}$, and honest parties are not allowed to adaptively choose their inputs) but rather will be supplied to both parties as a correlated input. See [5] for more details on how this step is implemented.

4. Conclusions

In this paper, we show how to construct the first concurrent non-malleable zero-knowledge protocol, assuming only that regular one-way functions exist. We also provide a new impossibility result regarding general functionalities, which together with [25, 26], gives us a better idea of where the border is between what is and is not possible in the plain model. An unfortunate consequence of the impossibility results is that we must move to alternative definitions of security for general functionalities if we want to obtain composable protocols for broader classes of functionality in the setting where there are no trusted parties or setup. One such definition was proposed in [35], by allowing super-polynomial time simulation. The main limitation of this definitional framework concerns functionalities whose *definitions* involve cryptographic primitives (or otherwise rely on computational complexity assumptions to be meaningful). For such functionalities, building on our techniques, one could hope to define and achieve security in a setting that a polynomial-time simulator is given extra powers, such as limited rewinding of the ideal model. (Of course, when relaxing security care must be taken that the definition still provides meaningful security guarantees for applications.) In fact, one may hope for a general clean definition that would provide the best of all worlds: for functionalities such as zero-knowledge provide full self composition, for functionalities where this is not possible provide some relaxed notions of security, and perhaps for functionalities that take as extra inputs a common reference string or input for a hard problem provide UC security or quasi-polynomial security. That is, there is hope for a clean meta-theorem from which

[7] The notations above assume that f is one-to-one, but this makes no difference in the proof.

Proceedings of the 47th Annual IEEE Symposium
on Foundations of Computer Science (FOCS'06)
0-7695-2720-5/06 $20.00 © 2006 IEEE

one could derive results such as [12, 6] and our current result by just plugging in the appropriate functionality.

References

[1] B. Barak. How to go beyond the black-box simulation barrier. In *Proc.* 42*nd FOCS*, pages 106–115, 2001.

[2] B. Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *Proc.* 43*rd FOCS*, 2002.

[3] B. Barak, R. Canetti, J. B. Nielsen, and R. Pass. Universally composable protocols with relaxed set-up assumptions. In *Proc.* 45*th FOCS*, pages 186–195, 2004.

[4] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *Crypto '01*, pages 1–18, 2001.

[5] B. Barak, M. Prabhakaran, and A. Sahai. Concurrent non-malleable zero knowledge. Cryptology ePrint Archive report., 2006. `http://eprint.iacr.org/`.

[6] B. Barak and A. Sahai. How to play almost any mental game over the net - concurrent composition using super-polynomial simulation. In *Proc.* 46*th FOCS*, 2005.

[7] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc.* 20*th STOC*, pages 1–10, 1988.

[8] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In B. Werner, editor, *Proc.* 42*nd FOCS*, pages 136–147, 2001. Preliminary full version available as Cryptology ePrint Archive Report 2000/067.

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

[10] R. Canetti, J. Kilian, E. Petrank, and A. Rosen. Black-box concurrent zero-knowledge requires (almost) logarithmically many rounds. *SIAM Journal on Computing*, 32(1):1–47, Feb. 2003. Preliminary version in STOC '01.

[11] R. Canetti, E. Kushilevitz, and Y. Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In *Eurocrypt '03*, 2003.

[12] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party computation. In *Proc.* 34*th STOC*, pages 494–503, 2002.

[13] I. Damgård, T. P. Pedersen, and B. Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. *J. Cryptology*, 10(3):163–194, 1997.

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

[15] C. Dwork, M. Naor, and A. Sahai. Concurrent zero knowledge. In *Proc.* 30*th STOC*, pages 409–418, 1998.

[16] J. A. Garay and P. D. MacKenzie. Concurrent oblivious transfer. In *Proc.* 41*st FOCS*, pages 314–324, 2000.

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

[18] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, Apr. 1988. Preliminary version in FOCS' 84.

[19] I. Haitner, O. Horvitz, J. Katz, C.-Y. Koo, R. Morselli, and R. Shaltiel. Reducing complexity assumptions for statistically-hiding commitment. In *EUROCRYPT*, pages 58–77, 2005.

[20] S. Halevi and S. Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *CRYPTO*, pages 201–215, 1996.

[21] Y. T. Kalai, Y. Lindell, and M. Prabhakaran. Concurrent general composition of secure protocols in the timing model. In *Proc.* 37*th STOC*, pages 644–653, 2005.

[22] J. Kelsey, B. Schneier, and D. Wagner. Protocol interactions and the chosen protocol attack. In *Proc. 1997 Security Protocols Workshop*, pages 91–104, 1997. Appeared in LNCS vol. 1361.

[23] J. Kilian and E. Petrank. Concurrent and resettable zero-knowledge in poly-logarithm rounds. In *Proc.* 33*th STOC*, pages 560–569, 2001. Preliminary full version published as cryptology ePrint report 2000/013.

[24] Y. Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *Proc.* 35*th STOC*, pages 683–692, 2003.

[25] Y. Lindell. General composition and universal composability in secure multi-party computation. In *Proc.* 44*th FOCS*, pages 394–403, 2003.

[26] Y. Lindell. Lower bounds for concurrent self composition. In *Theory of Cryptography Conference (TCC)*, volume 1, pages 203–222, 2004.

[27] T. Malkin, R. Moriarty, and N. Yakovenko. Generalized environmental security from number theoretic assumptions. In *TCC '05*, 2006.

[28] M. Naor, R. Ostrovsky, R. Venkatesan, and M. Yung. Perfect zero-knowledge arguments for p using any one-way permutation. *J. Cryptology*, 11(2):87–108, 1998.

[29] R. Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In *Eurocrypt '03*, 2003.

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

[31] R. Pass and A. Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *Proc.* 44*th FOCS*, 2003.

[32] R. Pass and A. Rosen. Concurrent non-malleable commitments. In *Proc.* 46*th FOCS*, 2005.

[33] R. Pass and A. Rosen. New and improved constructions of non-malleable cryptographic protocols. In *Proc.* 37*th STOC*, 2005.

[34] M. Prabhakaran, A. Rosen, and A. Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *Proc.* 43*rd FOCS*, 2002.

[35] M. Prabhakaran and A. Sahai. New notions of security: achieving universal composability without trusted setup. In *Proc.* 36*th STOC*, pages 242–251, 2004.

[36] R. Richardson and J. Kilian. On the concurrent composition of zero-knowledge proofs. In *Eurocrypt '99*, pages 415–432, 1999.

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

[38] A. C. Yao. How to generate and exchange secrets. In *Proc.* 27*th FOCS*, pages 162–167, 1986.

Proceedings of the 47th Annual IEEE Symposium
on Foundations of Computer Science (FOCS'06)
0-7695-2720-5/06 $20.00 © 2006 IEEE