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

October 21, 2006

Abstract

We provide the first construction of a concurrent and non-malleable zero knowledge argument for every language in $\mathbf{NP}$. 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.

Keywords: Non-malleable protocols, concurrent composition, concurrent zero knowledge, non-malleable zero knowledge

*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.

Contents

1 Introduction

In the two decades since their introduction [GMR85], 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 [GMR85] 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 his own, without any help from Alice. This property is called *non-malleability* [DDN91] 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 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 nontrivial 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* [DNS98, RK99], 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 $\mathbf{NP}$ 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 [NOVY98] and and even regular (and in particular one-to-one) one-way functions [HHK+05] and in constant rounds under claw-free permutations [GMR84] or collision-resistent hash functions [DPP97, HM96]. 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 [DDN91] and concurrent zero knowledge [DNS98, RK99], and is also similar to the analogous definitions for non-malleable and concurrent commitments [DDN91, PR05A]. We note that the best previous results on zero knowledge either (1) achieved only concurrent zero knowledge without non-malleability [RK99, KP01, PRS02], (2) achieved non-malleability but only with a bounded number of parties present [DDN91, BAR02, PR05B], (3) made use of global setup assumptions like a common reference strings [CLOS02] or time-delayed messages [KLP05], or (4) used different security frameworks like super-polynomial simulation [PS04, BS05, MMY06].

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 [LIN04]. 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 [CLOS02], bounded-concurrent security [LIN03A, PR03, PAS04], super-polynomial simulation [PS04, BS05, MMY06], and composition in timing model [KLP05]) 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 [RK99, DDN91] (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. [DDN91, PRS02, PR05A, BS05]) has been the development of new proof techniques to tame the complexity introduced by rewinding, often through new kinds of hybrid 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) [PRS02] concurrent zero knowledge protocol and simulation strategy. We also want to make use of non-malleable commitment constructions (e.g. [DDN91, PR05A]) 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 [DDN91, PR05A] 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

[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 [BAR01, BAR02, PAS03, PR03, PR05B, PR05A, BS05] 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 [BAR01] has not led to a constant-round concurrent zero-knowledge protocol.

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 [CAN01] or security against "chosen-protocol attack" [KSW97, LIN03B]) 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 [YAO86] 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 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 [BGI+01].)

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 [DNS98] and the first construction in the standard model was given by Richardson and Kilian [RK99]. The number of rounds was improved to $\tilde{O}(\log n)$ by [KP01, PRS02] which is optimal for *black-box simulation* [CKPR01]. (A constant round protocol for bounded-concurrent zero knowledge was given in [BAR01] using non-black-box simulation.)

Non-malleable zero knowledge. Non-malleable zero knowledge was first defined and constructed by Dolev, Dwork and Naor [DDN91]. Constant round protocols were given in [BAR02, PR05B]. These latter works also introduced some more convenient definitions (which we follow) than the [DDN91] definition (inspired by definitions of non-malleable *non-interactive* zero knowledge [SAH99]).

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 [DDN91] and constant-round protocols were given in [BAR02, PR05B]. Pass and Rosen [PR05A] showed that the commitment scheme from [PR05B] 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 [DDN91, CF01, CLOS02, LIN03A]. Thus, one might hope that one could obtain in this way a concurrent non-malleable ZK protocol from the [PR05A] scheme. However, an important limitation of [PR05A] 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 [PR05A]'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 [DDN91], 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 [Can01], 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 environment that contains executions of arbitrary other protocols as well (see also [Lin03b]). 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 [Can01, CF01, CKL03]. (See [BOGW88, Can01, CLOS02, BCNP04] for constructions in other models.)

Self-composition. As mentioned above, Lindell [Lin04] 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 [PS04, BS05, MMY06]. 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 [GM00] 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 $\texttt{Adv}$ 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 [DDN91] 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 $\texttt{Adv}$ 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 $\texttt{Adv}$ 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 $\mathcal{F}(x \circ w) = \bot$ otherwise (this functionality only uses one of its inputs).

2 A concurrent non-malleable zero knowledge protocol

Definition 2.1. A protocol is a Concurrent Non-Malleable Zero Knowledge (CNMZK) argument of knowledge for membership in an $\mathbf{NP}$ language L with witness relation R (that is, $y \in L$ iff there exists w such that $R(y, w) = 1$), if it is an interactive proof system between a prover and a verifier such that

Completeness: if both the prover and the verifier are honest, then for every (y, w) such that $R(y, w) = 1$, the verifier will accept the proof, and

Soundness, Zero-Knowledge and Non-Malleability: for every (non-uniform PPT) adversary $\mathcal{A}$ interacting with provers $P_1, \ldots, P_{m_L}$ in m_L "left sessions" and 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 simulator $\mathcal{S}$ such that for every set of "left inputs" $y_1, \ldots, y_{m_L}$, we have $\mathcal{S}(y_1, \ldots, y_{m_L}) = (\nu, z_1, \ldots, z_{m_R})$, such that

1. ν is a simulated view of $\mathcal{A}$: i.e., ν is distributed indistinguishably from the view of $\mathcal{A}$ (for any set of witnesses $(w_1, \ldots, w_{m_L})$ that $P_1, \ldots, P_{m_L}$ are provided with).[3]
2. For all $i \in \{1, \ldots, m_R\}$, if in the i-th right hand side session in ν the common input is x_i and the verifier V_i accepts the proof, then z_i is a valid witness to the membership of x_i in the language, except with negligible probability ($z_i = \bot$ if V_i does not accept.)

Further, we call the protocol a black-box CNMZK if there exists a universal simulator $\mathcal{S}_{\text{BB}}$ such that for any adversary $\mathcal{A}$, it is the case that $\mathcal{S} = \mathcal{S}_{\text{BB}}^{\mathcal{A}}$ satisfies the above requirements.

Note that the above security property subsumes both zero-knowledge and proof-of-knowledge properties. In particular, the second condition reduces to regular (stand-alone) zero knowledge property when $m_L = 1$ and $m_R = 0$, and implies regular (stand-alone) proof-of-knowledge property when $m_L = 0$ and $m_R = 1$. Furthermore, this condition reduces to concurrent zero knowledge [DNS98, RK99] when $m_L = poly$ and $m_R = 0$; it reduces to basic ("non-concurrent") non-malleability [DDN91] when $m_L = m_R = 1$.

This definition resembles the notion of simulation-extractablity used in [PR05A] for concurrent non-malleable commitments.

[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.

2.1 UC-like definition of CNMZK

We can also write this definition in the language of the UC-framework, to further illustrate the level of security and composition it gives. We do not get into the details of modeling the Network, but instead keep our description at an informal level. For more details of modeling see [Can05, Pra05].

The functionality in question is $\mathcal{F}_{\text{ZK}}^{R}$, the natural zero-knowledge functionality for membership in L: it accepts a pair (y, w) from P and sends $(y, R(y, w))$ to V, which it outputs.

The nature of the security is essentially described by the kind of environments allowed in the security definition. We call a PPT environment $\mathcal{Z}$ a "CNM environment" if it behaves as follows:

- $\mathcal{Z}$ interacts arbitrarily with the adversary, and selects many pairs of parties (P, V). For each of these pairs, $\mathcal{Z}$ picks (y, w) such that $R(y, w) = 1$. It hands y to both P and V, and w to P.
- Then $\mathcal{Z}$ initiates each pair to interact with an instance of $\mathcal{F}_{\text{ZK}}^{R}$. After this point the environment does not send any messages to the parties or to the adversary (but it continues to receive messages from the adversary).[4]
- Finally $\mathcal{Z}$ outputs a bit.

Note that since there will be no automatic composition theorem available, the environment already invokes multiple instances of the functionality. Also note that there are no other protocols or functionalities being invoked, emphasising the fact that we are dealing only with self-composition.

In the "ideal" execution, when initiated with input, the parties interact with $\mathcal{F}_{\text{ZK}}^{R}$. In the "real" execution the parties use the protocol in question. All scheduling is controlled adversarially.

Then the definition of security is that there exists a simulator $\mathcal{S}_{\text{BB}}$ such that for all adversaries $\mathcal{A}$ and *any CNM environment* $\mathcal{Z}$, the output of $\mathcal{Z}$ in the real execution is indistinguishable from that in the ideal execution.

2.2 Result from [PRS02]

We adapt the main argument from [PRS02] for use in our protocol. Consider the following protocol segment:[5]

1. **PRS Commitment:** The verifier picks a (sufficiently long) random string σ, and prepares $k \cdot t(k)$ (where $t(k)$ is any $\omega(\log k)$ function) pairs of secret shares $(\alpha_{ij}^{0}, \alpha_{ij}^{1})$ for $1 \le i \le k$, $1 \le j \le t(k)$ such that for all (i, j) we have $\alpha_{ij}^{0} \oplus \alpha_{ij}^{1} = \sigma$. Then it commits to σ and $(\alpha_{ij}^{0}, \alpha_{ij}^{1})_{ij}$ using a statistically binding commitment scheme $\mathsf{Com}_{\mathsf{PRS}}$.
2. **PRS Challenge-Response:** This is followed by $t(k)$ rounds of random k-bit challenges by the prover. In response, for each (i, j), if the i-th bit in the j-th challenge $r_{ij} = b$ then the verifier opens the commitment to α_{ij}^{b} in that round.
3. On reaching this point the prover considers the preamble to have "concluded."

[4]The environment not communicating with honest parties after initiating the protocol execution corresponds to the restriction that the inputs to the honest parties are non-adaptive. Also the restriction that there is no information the adversary receives other than its initial auxiliary inputs and the protocol messages is captured by restricting the environment from sending messages to the adversary after the protocols commence. Technically, such a restricted environment allows a simulation that involves "rewinding," as in our case.

[5]In describing all protocols we use the implicit convention that if a party receives messages which are not as prescribed by the protocol, that party is required to abort the entire session.

4. **PRS Opening:** The verifier opens all the commitments made in the PRS Commitment step, and the prover verifies consistency of the revealed values (i.e., that for all (i, j), $\alpha^0_{ij} \oplus \alpha^1_{ij}$ evaluates to the same value).

5. On reaching this point the prover "accepts" the preamble.

There can be other messages in between, as long as the challenges r_{ij} are picked randomly independent of previous messages. In particular, as in our case, there can be 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, $\mathcal{A}_{\text{PRS}}$. The simulator gets inputs of all the parties in all the sessions, and it runs the honest provers and the adversarial verifier internally.[6] In the end it produces an ordered list of "threads of execution." A thread of execution consists of views 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 of the session 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." [PRS02] guarantees that the probability of the PRS simulation getting stuck is negligible.

Lemma 2.2. *(Adapted from [PRS02]) 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.*[7]

In fact [PRS02] prove a further refinement of this lemma (that we 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,[8] the lemma still holds. (This is what allows [PRS02] to show that indistinguishable

[6] 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.

[7] Unlike in [PRS02], in our preamble, the PRS commitment is statistically binding. So, except with negligible probability, if P_i accepts the preamble, there is a well-defined value σ in the PRS commitment, and it is this value that the prover accepted as the secret in the preamble. We point out that our case is slightly simpler than the original analysis in [PRS02] in that we are interested in arguments (not proofs), and hence the commitment by the verifier can be statistically binding.

[8] In our applications, it is enough if this holds when the indistinguishability is statistical; but in fact this refinement holds even if the indistinguishability is only computational. Indeed in [PRS02] the argument is used for computationally indistinguishable executions.

simulation is possible.). 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. We could build a preliminary simulator (which is provided with inputs of all the parties) for this situation by considering all the right hand verifiers also as part of an adversary $\mathcal{A}_{\text{PRS}}$ before invoking the PRS simulator. However there is a minor technicality that needs to be taken into account. In [PRS02], since the adversary is arbitrary, it may very well be assumed to read its entire random tape up front. Thus in all the threads (all of which may share a common non-empty prefix) the PRS simulator in [PRS02] uses the same random tape for the adversary. But it is easy to see that the analysis in [PRS02] works even with probabilistic adversaries which do not read their entire random tapes initially, and in that case the PRS simulator can use fresh randomness for the unread parts of the random tape when simulating a new thread. 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). So in our use of the PRS simulator only the random tape of the original (arbitrary) adversary $\mathcal{A}$ is fixed across all the threads while the rest of $\mathcal{A}_{\text{PRS}}$ (i.e., the right hand side verifiers) is given fresh randomness in different threads.

Another equivalent (and in some sense a more natural) way to formulate this is to consider the right hand side verifiers as part of the honest prover and, as in the original PRS simulator, to fix the random tape of the adversary across all threads. Later in our proof, we will have chance to refer to this formulation.

2.3 Non-Malleable Commitment

The other ingredient we need is a statistically binding non-malleable commitment (not necessarily concurrent non-malleable) with an "extractability" property. More precisely, we require an interactive commitment protocol $\mathsf{Com}_{\mathsf{NM}}$, between a "sender" (whose input it wants to commit to) and a "receiver" (with no input) satisfying the following properties.

1. **Statistical Binding:** The protocol has a determining message from the sender to the receiver (typically the first message from the sender) which is the first message containing information about the value to be committed. If either the sender or the receiver is honest, the determining message is information theoretically binding except with negligible probability.

   For clarity in presentation we shall require that the first message in the protocol is itself the determining message. (However see Section 2.5.2.)

2. **(Non-concurrent) Non-Malleability:** Consider the following two experiments in which an adversary M participates in one "left session" of the protocol as the receiver, and in one "right session" as the sender. M picks a value w and gives it to the left sender P. In the first experiment P commits to w while in the second experiment it commits to the all-zeroes string. We define the value of the experiment as (τ, α), where τ is the output of M and α is the value in the determining message of the right-hand-side commitment. (We say $\alpha = \perp$ if the right-hand-side commitment was not accepted, or if the determining message did not uniquely determine the committed value).

   The non-malleability property is that the values of the two experiments are distributed computationally indistinguishably.

   For the sake of convenience we state the hiding property explicitly, though it is implied by the non-malleability property. The two experiments are defined as before, except that M does not participate in a right hand execution. Instead, after receiving the commitment from P, M produces an output, which is the value of the experiment. Then the hiding property requires that the values of the two experiments are computationally indistinguishably distributed.

3. **(Stand-alone) Extractability:** The extractability requirement is that there is an efficient extractor such that given an adversary and its view from a random execution of the protocol with an honest receiver, then, except with negligible probability – the probability being over the coins of the adversary and the verifier in the view, as well as that of the extractor, if it is randomized – the extractor outputs the value in the commitment, if according to the view the receiver accepts the commitment.

   In fact, we require a slight extension to this by requiring that the extraction can work on a prefix of the protocol where the verifier is public-coin. More formally, there is a message from the sender in the protocol called the "knowledge-determining message" (KDM), such that given an adversary and its view during a random execution of the protocol till (and including) the KDM, the extractor will output the committed value, if according to the view the verifier was still accepting (i.e., it did not abort). We require that prior to receiving the KDM the receiver does not have any private coins.

2.3.1 Available Non-Malleable Commitment Protocols

For simplicity, first we consider a model in which all parties have distinct identities[9], and all communication is over authenticated channels.

Pass-Rosen Commitment: In [PR05A] give a statistically binding commitment protocol, and show that their protocol satisfies the non-malleability property we require (and more).

Though [PR05A] states their definition without an extraction requirement, in their proof they show how to do extraction as well. But in fact we observe that another simple extractor (so that the protocol is clearly public coin until the knowledge-determining message) can be derived by slightly modifying their protocol. Their protocol has the following structure: first the sender commits to its input using *any* statistically binding commitment scheme; then it gives a proof of knowledge of the input and randomness used in this commitment, using a non-malleable ZK protocol nmZK_{ID}, where ID is the identity of the prover. Consider modifying this protocol by replacing the first message – namely the statistically binding commitment – by an *interactive* statistically binding commitment, which consists of a regular non-interactive statistically binding commitment (which can be based on any 1-1 one-way function) followed by a ZK proof of knowledge that it knows the contents of the commitment. Using a ZKPOK of super-constant rounds we can obtain a deterministic polynomial time extractor. This can be done, for instance, by having a $\omega(logk)$-round sequential copies of the basic Hamiltonicity protocol [BLU87]. (see e.g. [GOL01]).

This commitment protocol satisfies all our requirements.

DDN Commitment: Surprisingly, though we are in a concurrent setting, our requirement of non-malleability on the commitment scheme $\mathsf{Com}_{\mathsf{NM}}$ is in the plain non-malleability setting (i.e., one execution each on the left hand and right hand sides). We show that the original non-malleable commitment scheme by Dolev, Dwork and Naor [DDN91] satisfies our requirements, when the initial (statistically binding, non-interactive)

[9] This assumption can be removed (as originally done in [DDN91]) by letting the honest parties pick a (signing key, verfication key)-pair for a signature scheme, and having the entire transcript of the protocol signed using this key (only the provers need to sign). Then the one case excepted from the definition of non-malleability is when $\mathcal{A}$ copies an entire left execution as a right execution, by playing a router for the messages.

commitment phase of their protocol is augmented to have a ZKPOK, as above. As we mentioned above this modification takes care of the extraction requirement, while retaining the non-malleability property proven there.

Next we claim that the non-malleability property of the DDN protocol implies the non-malleability property that we require. First we *adapt* the definition from [DDN91] as follows:

Definition 2.3. (**DDN Non-Malleable Commitment. Adapted from [**DDN91**].**) A statistically binding commitment protocol is said to be DDN-non-malleable if for every (non-uniform PPT) adversary $\mathcal{A}_{\text{DDN}}$, there exists a simulator $\mathcal{S}_{\text{DDN}}$ such that for all (non-uniform) PPT machines $\mathcal{R}$ that take three inputs and outputs a single bit, the outputs of the following two experiments are indistinguishable from each other.

DDN-Experiment 1: $\mathcal{A}_{\text{DDN}}$ first outputs a string w. Then a is set to be w or the all-zeros string, chosen unifromly at random. An honest sender P commits to a to $\mathcal{A}_{\text{DDN}}$. Meanwhile $\mathcal{A}_{\text{DDN}}$ commits to some string α to an honest receiver V. (α is well-defined, possibly as $\perp$, because the commitment is statistically binding). Also at the end $\mathcal{A}_{\text{DDN}}$ outputs a plain-text τ_{DDN}. The output of the experiment is $\mathcal{R}(a, \alpha, \tau_{\text{DDN}})$.

DDN-Experiment 2: $\mathcal{S}_{\text{DDN}}$ outputs a string w and a is set to be w or the all-zeros string, chosen unifromly at random. Then $\mathcal{S}_{\text{DDN}}$ commits to some string α to an honest receiver V and outputs a plain-text τ_{DDN}. The output of the experiment is $\mathcal{R}(a, \alpha, \tau_{\text{DDN}})$.

Here we have simplified the DDN definition by replacing a general distribution by a uniform distribution over a string w and the all-zeros string. Without loss of generality we assume that w is chosen deterministically (but non-uniformly). Also, we have slightly strengthened the definition to include a plain-text output τ produced by the adversary as input to $\mathcal{R}$. To see that the protocol in [DDN91] does satisfy this strengthened requirement, note that the proof there first uses a "knowledge extractor" to extract α, which could be modified to output (τ, α) instead.

Now suppose a commitment protocol did not satisfy the non-malleability we require. Then, there is an adversary $\mathcal{A}$ who gives a value w, accepts a commitment on the left to either w or the all-zero string, and makes a commitment on the right to computationally distinguishable values on the right, and outputs a string τ. Let the value committed to on the right be α. Denote these as $(\tau^{(w)}, \alpha^{(w)})$ in the first experiment (when commitment on the left is to w), and as $(\tau^{(0)}, \alpha^{(0)})$ in the second experiment (when commitment on the left is to 0). We have a PPT distinguisher D which outputs a single bit such that $D(\tau^{(w)}, \alpha^{(w)}) \not\equiv_{\text{C}} D(\tau^{(0)}, \alpha^{(0)})$. In other words, $|\pi_w - \pi_0|$ is not negligible, where π_w and π_0 stand for $\Pr[D(\tau^{(w)}, \alpha^{(w)}) = 1]$ and $\Pr[D(\tau^{(0)}, \alpha^{(0)}) = 1]$ respectively. Note that this can be true only if w is not the all-zeros string. Now define $\mathcal{A}_{\text{DDN}}$ to be the same as $\mathcal{A}$, but with $\tau_{\text{DDN}} = (\tau, w)$. Define

$$\mathcal{R}(a, \alpha, (\tau, w)) = \begin{cases} D(\alpha, \tau) & \text{if } a = w \\ 1 - D(\alpha, \tau) & \text{otherwise.} \end{cases}$$

Then the probability of the output of DDN-Experiment 1 being 1 is $\frac{1}{2}(\pi_w + (1 - \pi_0)) = \frac{1}{2} + \frac{1}{2}(\pi_w - \pi_0)$, where as the probability of DDN-Experiment 2 producing 1 is $\frac{1}{2}$ (since $a = w$ with probability $\frac{1}{2}$; here we use the fact that w is not the all-zeros string). Since $|\pi_w - \pi_0|$ is not negligible, the outputs of the two experiments are not indistinguishable. Hence we conclude that if the protocol is not non-malleabile in the form we require, then it is not DDN-non-malleable either.

2.4 Other Ingredients

The other ingredients we use are

1. A statistically (or perfectly) hiding commitment scheme $\mathsf{Com}_{\mathsf{SH}}$.
2. A statistical (or perfect) ZK argument of knowledge sZKAOK, for proving knowledge of witness for membership in any $\mathbf{NP}$ language.

The statistically hiding commitment scheme $\mathsf{Com}_{\mathsf{SH}}$ can be achieved in a constant-round protocol using collision-resistant hash functions or claw-free permutations or, at the expense of having $O(k)$ rounds, using one-way permutations [NOVY98] (see also Section 4.8 of [Gol01]), and even using only regular one-way functions by the recent result of Haitner *et al.*[HHK+05]. Given such a commitment scheme, we get sZKAOK as required with a factor $\omega(\log k)$ blow up in the number of rounds, in same manner as our construction of a ZKPOK above, using $\omega(\log k)$ sequential copies of the Hamiltonicity protocol, but where the prover's commitments in the Hamiltonicity protocol are made using the statistically hiding commitment scheme $\mathsf{Com}_{\mathsf{SH}}$. This sZKAOK enjoys a strict polynomial-time extraction procedure with negligible probability of failure.

We note that all our ingredients are realizable under the assumption that regular one-way functions exist (and in particular under the assumptions that one-to-one one-way functions exist).

2.5 Our Protocol

Consider an $\mathbf{NP}$-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 from Section 2.2 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 2.4. *Protocol* CNMZK *is a black-box concurrent non-malleable zero knowledge argument for membership in the* $\mathbf{NP}$ *language* L *(Defintion 2.1).*

Proof. It is easy to see that the protocol satisifies the completeness condition. Below we shall build a simulator-extractor, which outputs a simulated view of the adversary's view along with witnesses for all the successful right hand side proofs in the simulated view, as required by the second condition in Defintion 2.1.

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 (which is statistically binding) in that session indeed contains a valid witness z_ℓ to the statement x_ℓ. (This follows from a hybrid argument for the m_R right hand side sessions.) This is stated in the claim below; a detailed proof follows.

Claim 2.5.

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

except with negligible probability.

Proof. Fix $\ell \in \{1, \ldots, m_R\}$. First, from $\mathcal{H}_1$, construct a standalone prover P^* which interacts with V_ℓ alone. This is done by including everything simulated by $\mathcal{H}_1$ except V_ℓ as part of P^*, so that an interaction of P^* with an honest verifier V_ℓ is identical to the execution of $\mathcal{H}_1$. We need to argue that if V_ℓ accepts the proof by P^*, then except with negligible probability, the Phase IV commitment made by P^* is a valid witness z_ℓ to the statement x_ℓ. We consider the following experiment. Engage P^* in an execution with an honest verifier V_ℓ which uses a random PRS secret σ. Then, if V_ℓ accepts the proof from P^* build two standalone provers P_1^* and P_2^* as follows. P_1^* is a copy of P^* at the point where it began the sZKAOK in Phase II. P_2^* is a copy of P^* at the point where it began the sZKAOK in Phase V. Now run the extractor for sZKAOK on P_1^* and P_2^*. First we observe that from the hiding property of $\mathsf{Com}_{\mathsf{PRS}}$ it follows that the probability of the extractor on P_1^* returning (an opening or explanation of $\mathsf{Com}_{\mathsf{SH}}$ as a commitment to) σ is negligible. Secondly we observe that the computational binding of $\mathsf{Com}_{\mathsf{SH}}$ implies that the probability of extractor on P_1^* returning an opening to something other than σ *and* the extractor on P_2^* returning an opening to σ is negligible: this is because, otherwise we obtain two different ways to open $\mathsf{Com}_{\mathsf{SH}}$. Finally by the knowledge extractability property of sZKAOK we observe that the probability of (V_ℓ accepting and) P_1^* not returning some opening of $\mathsf{Com}_{\mathsf{SH}}$ is negligible; also that of P_2^* returning neither an opening of $\mathsf{Com}_{\mathsf{SH}}$ to σ nor an opening of the Phase IV $\mathsf{Com}_{\mathsf{NM}}$ commitment to a valid witness for x_ℓ is negligible. Together these imply that the probability of V_ℓ accepting the proof and the Phase IV $\mathsf{Com}_{\mathsf{NM}}$ being not to a valid witness for x_ℓ is negligible. □

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 $\nu^{(2)}$. If the simulator did not check for the aborting condition, the view generated is identically distributed as in the simulation by $\mathcal{H}_1$. By Lemma 2.2 we know that the probability of aborting is negligible. Hence, we have

$$\nu^{(1)} \equiv_{\mathrm{C}} \nu^{(2)}$$
$$\forall \ell \qquad (b_\ell^{(1)}, \alpha_\ell^{(1)}, y_\ell) \equiv_{\mathrm{C}} (b_\ell^{(2)}, \alpha_\ell^{(2)}, y_\ell).$$

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

$$\nu^{(2)} \equiv_{\mathrm{C}} \nu^{(3)}$$
$$\forall \ell \qquad (b_\ell^{(2)}, \alpha_\ell^{(2)}, y_\ell) \equiv_{\mathrm{C}} (b_\ell^{(3)}, \alpha_\ell^{(3)}, y_\ell).$$

Stage 4: $\mathcal{H}_4$ 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_ℓ.

Claim 2.6.

$$\nu^{(3)} \equiv_C \nu^{(4)}$$
$$\forall \ell \qquad (b_\ell^{(3)}, \alpha_\ell^{(3)}, y_\ell) \equiv_C (b_\ell^{(4)}, \alpha_\ell^{(4)}, y_\ell).$$

Here $\nu^{(4)}$ and $\{b_\ell^{(4)}, \alpha_\ell^{(4)}\}_{\ell=1}^{m_R}$ are defined analogously to the case of $\mathcal{H}_1$. We shall prove this claim shortly, using a carefully designed series of hybrids. It is in this part of the proof that we shall require the non-malleability property of the commitment scheme $\mathsf{Com}_{\mathsf{NM}}$.

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 this we take the view that in the PRS simulator all the honest parties including V_ℓ are considered part of the prover. Then, for each ℓ, $\mathcal{S}$ will consider $\mathcal{H}_4$ as a standalone adversary $\mathcal{A}_\ell^*$ making a single commitment to a receiver. The adversary $\mathcal{A}_\ell^*$ will contain the adversary and all of the honest parties simulated by $\mathcal{H}_4$, except the part of V_ℓ in the main thread which receives the Phase IV commitment $\mathsf{Com}_{\mathsf{NM}}$. $\mathcal{A}_\ell^*$ terminates execution after sending the knowledge-determining message (KDM) to the external verifier.

Note that some of the PRS look-ahead threads simulated by $\mathcal{H}_4$ will share a prefix with the main thread. Thus the interaction of $\mathcal{A}_\ell^*$ with the external receiver (which forms part of the main thread) may define parts of these look-ahead threads as well. If the KDM to V_ℓ in the main thread does not occur in the shared prefix with a look-ahead thread, then $\mathcal{H}_4$ would have created this thread before reaching the KDM. Hence $\mathcal{A}_\ell^*$ also needs to create this thread before terminating. For simulating such a look-ahead thread which shares some prefix with the interaction with the external receiver, $\mathcal{A}_\ell^*$ should be able to internally *continue* a prefix of the interaction with an external receiver where the prefix does not extend to the KDM. This is possible because of our requirement that prior to the KDM the receiver in $\mathsf{Com}_{\mathsf{NM}}$ does not use any private coins. So at the point $\mathcal{A}_\ell^*$ needs to continue this prefix as a look-ahead, there is no secret state of the receiver that it needs to know. It simply continues the look-ahead thread using fresh coins for the verifier. Thus $\mathcal{A}_\ell^*$ is indeed well-defined.

$\mathcal{S}$ constructs the view of $\mathcal{A}_\ell^*$ (by having kept track of the internals in the run of $\mathcal{H}_4$) and invokes the extractor for $\mathsf{Com_{NM}}$, with $\mathcal{A}_\ell^*$ and this view. The final output of $\mathcal{S}$ is $(\nu, \beta_1, \ldots, \beta_{m_R})$ where β_ℓ are the extracted values. 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.

Note that from above displayed relations, $\nu^{(4)} \equiv_{\mathrm{C}} \nu^{(1)}$, 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

$$\forall \ell \qquad \left(b_\ell^{(4)} = 1\right) \implies \left(R(x_\ell, \alpha_\ell^{(4)}) = 1\right)$$

except with negligible probability. This follows from Equation 1, the fact that $(b_\ell^{(4)}, \alpha_\ell^{(4)}) \equiv_{\mathrm{C}} (b_\ell^{(1)}, \alpha_\ell^{(1)})$ as implied by the above displayed relations, 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 2.6. $\square$

2.5.1 Proof of Claim 2.6

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. To describe these hybrids, first we introduce some notation. In the PRS simulation consider numbering (in order) all the occurrences of 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 $p(i)$.

Consider a FM and all threads passing through it (i.e., all threads which share a prefix containing this FM). Suppose this FM belongs to a left hand session with prover P_j. In each of the threads, the session with P_j may go on to reach Phase V sZKAOK. We will refer to these instances of sZKAOK as "belonging" to this particular FM.

Now we can describe our intermediate hybrids $\tilde{\mathcal{H}}_{i:1}$ and $\tilde{\mathcal{H}}_{i:2}$. 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 ($N = O((m_L t(k))^2)$ suffices). 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

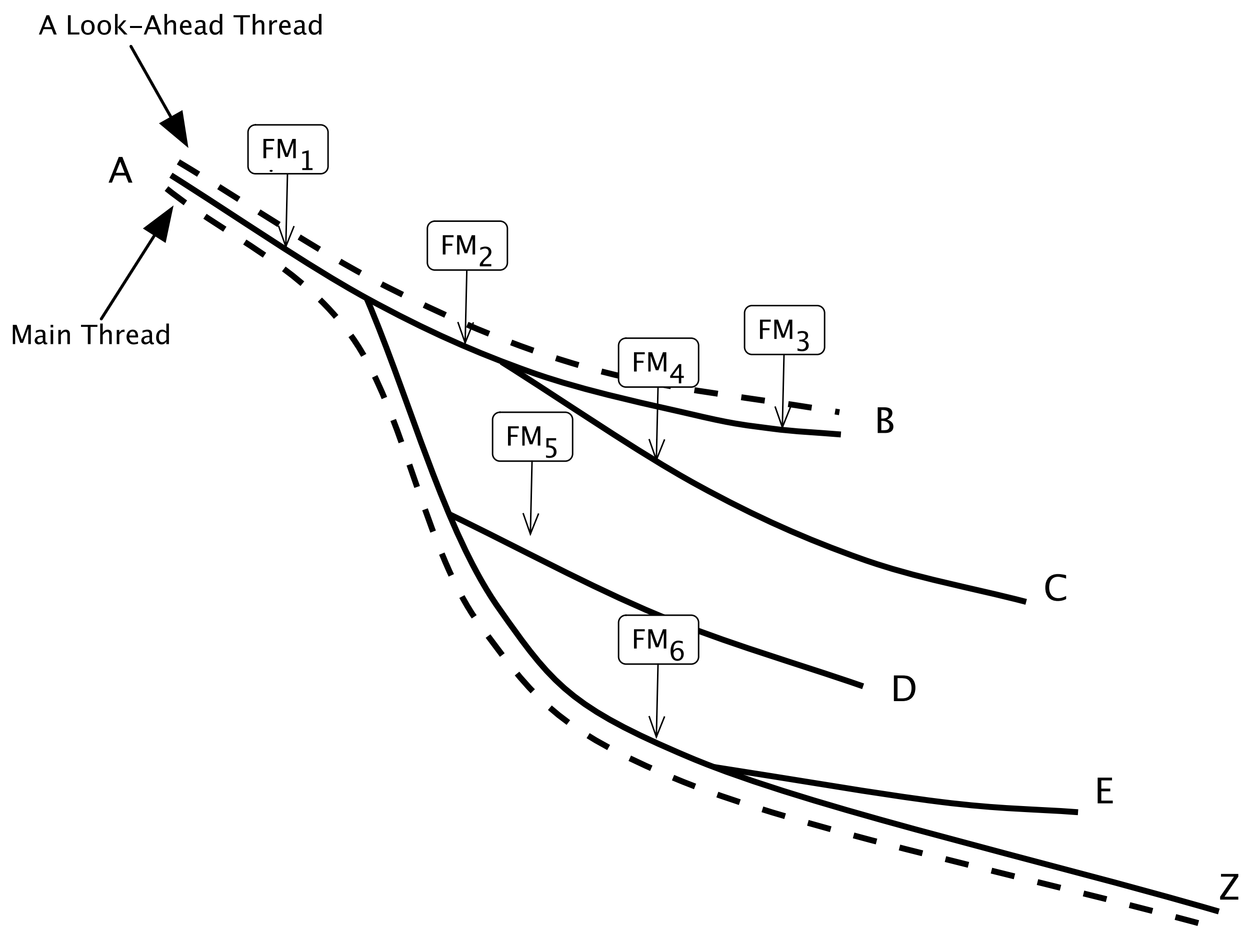

Figure 1: A schematic representation of the threads in a PRS simulation. Here the segment AB represents the first thread and AZ the last or main thread (highlighted with dotted lines). AB, AC, AD, AE are all look-ahead threads. Also marked are points where the FMs (first messages of Phase IV $\mathsf{Com}_{\mathsf{NM}}$ from $\mathcal{A}$ to the right hand side verifiers) occurred during this simulation. Note the order in which FMs are numbered.

the simulator fails[10].

$\tilde{\mathcal{H}}_{i:2}$**:** 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

$$\tilde{\nu}^{(0:2)} \equiv_{\mathrm{C}} \tilde{\nu}^{(N:2)}$$

$$\forall \ell \qquad (\tilde{b}_\ell^{(0:2)}, \tilde{\alpha}_\ell^{(0:2)}, y_\ell) \equiv_{\mathrm{C}} (\tilde{b}_\ell^{(N:2)}, \tilde{\alpha}_\ell^{(N:2)}, y_\ell).$$

We do this via the following sequence:

$$\tilde{\nu}^{(i-1:2)} \equiv_{\mathrm{C}} \tilde{\nu}^{(i:1)} \tag{2}$$

$$\tilde{\nu}^{(i:1)} \equiv_{\mathrm{C}} \tilde{\nu}^{(i:2)} \tag{3}$$

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

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

Proving Equations (2) **and** (4)**:** These follow from the fact that the Phase V sZKAOK remains statistically indistinguishable when the alternate witness is used. However note that in the PRS simulation, 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. Recall the refinement mentioned after Lemma 2.2: as the change introduced in each thread is undetectable, it will still hold that the PRS secrets will be available as required except with negligible probability. Other than the availability of the PRS secret, each thread is independent of other threads. Thus each individual thread, and in particular the main thread, continues to be statistically indistinguishable in the simulation by $\tilde{\mathcal{H}}_{i-1:2}$ and $\tilde{\mathcal{H}}_{i:1}$. This in turn implies both equations (2) and (4).

Proving Equations (3) **and** (5)**:** Equation (3) follows from the hiding property of $\mathsf{Com}_{\mathsf{NM}}$. To see this we create a standalone machine M which is identical to $\tilde{\mathcal{H}}_{i:1}$, except that on reaching FM_i it starts interacting with an external sender P. First it sends $w_{p(i)}$ to P, and then receives a commitment from P which it uses to interact with $\mathcal{A}$, instead of an honest commitment to $w_{p(i)}$ as $\tilde{\mathcal{H}}_{i:1}$ does. However if P makes an honest commitment to $w_{p(i)}$ then the (M, P) system is identical to $\tilde{\mathcal{H}}_{i:1}$. However, if P makes a commitment to the all-zeros string, then the (M, P) system is identical to $\tilde{\mathcal{H}}_{i:2}$. By the hiding property of $\mathsf{Com}_{\mathsf{NM}}$ the output of M must be indistinguishable in the two cases, establishing equation (3).

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 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 $\tilde{\mathcal{H}}_{i:2}$.

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

The precise argument is slightly more involved. Consider generating the pair of random variables $(\tilde{b}_\ell^{(i:1)}, \tilde{\alpha}_\ell^{(i:1)})$ and $(\tilde{b}_\ell^{(i:2)}, \tilde{\alpha}_\ell^{(i:2)})$ as follows: note that $\tilde{\mathcal{H}}_{i:1}$ and $\tilde{\mathcal{H}}_{i:2}$ are described identically until FM_i. Let us call this machine $\tilde{\mathcal{H}}_i$, which we run until reaching FM_i. At this point there are three possibilities:

1 Both $(\tilde{b}_\ell^{(i:1)}, \tilde{\alpha}_\ell^{(i:1)})$ and $(\tilde{b}_\ell^{(i:2)}, \tilde{\alpha}_\ell^{(i:2)})$ are defined – i.e. the ℓ'th right-hand-side interaction already terminated before we reached FM_i. In this case they have identical values.

2 Both $\tilde{\alpha}_\ell^{(i:1)}$ and $\tilde{\alpha}_\ell^{(i:2)}$ are defined and they have the same value, $\tilde{\alpha}_\ell^{(i)}$. $\tilde{b}_\ell^{(i:1)}$ and $\tilde{b}_\ell^{(i:2)}$ are not yet defined.

3 All of $(\tilde{b}_\ell^{(i:1)}, \tilde{\alpha}_\ell^{(i:1)})$ and $(\tilde{b}_\ell^{(i:2)}, \tilde{\alpha}_\ell^{(i:2)})$ are undefined.

In the latter two cases we continue the execution of $\tilde{\mathcal{H}}_{i:1}$ and $\tilde{\mathcal{H}}_{i:2}$ separately to fully define the random variables. It is sufficient to argue that the pairs of random variables obtained in these two sub-processes are indistinguishably distributed.

In the second case this follows from the indistinguishability of $\mathsf{Com}_{\mathsf{NM}}$. To see this consider a machine M_ℓ initialized to $\tilde{\mathcal{H}}_i$, at the point of FM_i. It starts off by sending $w_{p(i)}$ to an external sender P and receives a commitment to either $w_{p(i)}$ or to the all zeros string, and uses this commitment instead of the honest commitment; then it outputs a bit indicating whether V_ℓ accepted the simulated proof or not, along with the committed value $\tilde{\alpha}_\ell^{(i)}$, which is provided to M_ℓ non-uniformly (as it was already fixed). Depending on the choice made by P, the output is either $(\tilde{b}_\ell^{(i:1)}, \tilde{\alpha}_\ell^{(i:1)})$ or $(\tilde{b}_\ell^{(i:2)}, \tilde{\alpha}_\ell^{(i:2)})$. Since M is a (non-uniform) PPT machine, the hiding property of $\mathsf{Com}_{\mathsf{NM}}$ implies that these two outputs must be indistinguishable.

In the third case we consider two sub-cases, depending on whether FM_i is in the main thread or not. If FM_i is not in the main thread, then the statistical difference in the main thread is negligible. This is because, as described in the proof of equations (2) and (4), the only way previous threads affect the main thread is on account of whether all the requisite PRS secrets are available on time during the simulation; but, again as pointed out above, the refinement of Lemma 2.2 guarantees that the probability of the simulation getting stuck for want of a PRS secret will remain negligible in both $\tilde{\mathcal{H}}_{i:1}$ and $\tilde{\mathcal{H}}_{i:2}$.

Dealing with the remaining sub-case, when FM_i appears in the main thread, requires the non-malleability property of $\mathsf{Com}_{\mathsf{NM}}$. Note that at FM_i the first message of the right hand $\mathsf{Com}_{\mathsf{NM}}$ phase with V_ℓ has not yet started. Then, like before, we construct a machine M'_ℓ, again initialized to $\tilde{\mathcal{H}}_i$ at the point of FM_i, which starts off by sending $w_{p(i)}$ to an external sender P and receives a commitment to either $w_{p(i)}$ or to the all zeros string, and uses this commitment instead of the honest commitment. However M'_ℓ differs from M_ℓ in the following ways:

- It does not run any look-ahead threads. (For instance, in Figure 1, for $i = 1$, M'_ℓ will not run any look-ahead threads; for $i = 6$, M'_ℓ will run all the look-ahead threads except AE.) Note that at the point FM_i, we have that $\tilde{\mathcal{H}}_i$ would have recorded the PRS secret for all the left hand side preambles which were already concluded and could go on to be accepted. As for the preambles which are concluded after the point FM_i, if their sessions go on to reach Phase IV $\mathsf{Com}_{\mathsf{NM}}$, then they will be numbered FM_j for $j \geq i$. So for those sessions $\tilde{\mathcal{H}}_{i:1}$ and $\tilde{\mathcal{H}}_{i:2}$ do not require the PRS secret to execute the thread. So there is no need to run any further look-ahead threads.

- The $\mathsf{Com}_{\mathsf{NM}}$ commitment to V_ℓ is "exposed." That is, the part of V_ℓ which receives the $\mathsf{Com}_{\mathsf{NM}}$ commitment is not internalized; instead M'_ℓ expects this to be an outside party.

- For convenience, we will have M'_ℓ output a bit indicating whether in the internal simulation V_ℓ accepted the proof or not.

When P chooses to commit to the string sent by M'_ℓ, the entire execution, with an honest external receiver for the exposed commitment is a statistical simulation of the main thread execution of $\tilde{\mathcal{H}}_{i:1}$, and when P chooses to commit to the all-zeros string it is a statistical simulation of the main thread execution of $\tilde{\mathcal{H}}_{i:2}$. (The only reason for the simulations being not perfect is that in M'_ℓ, the negligible probability that the PRS simulation may fail beyond FM_i is no longer present, whereas it is present in $\tilde{\mathcal{H}}_{i:1}$ and $\tilde{\mathcal{H}}_{i:2}$.) Further, in the first case the output by M'_ℓ is $\tilde{b}_\ell^{(i:1)}$ and in the latter $\tilde{b}_\ell^{(i:2)}$. Now, the non-malleability condition on $\mathsf{Com}_{\mathsf{NM}}$ implies equation (5). □

2.5.2 Relaxing the requirement on $\mathsf{Com}_{\mathsf{NM}}$

We remark that the (natural) requirement we used, that the first message in $\mathsf{Com}_{\mathsf{NM}}$ be the determining message, can in fact be removed. This will provide the flexibility of using a protocol based on alternate statistical binding commitment schemes (like Naor's scheme [Nao89] in which the first message in the protocol is a random string from the receiver to the sender, and it is the second message which is the determining message).

Not having the first message as the determining messageg affects our proof at exactly one point:[11] at the very last case analysis in the proof of Claim 2.6, in constructing the adversary M'_ℓ we assumed that it can be initialized to the point at which FM_i occurs in the main thread of $\tilde{\mathcal{H}}_i$, and only subsequently does it start interacting with the external receiver, in the exposed session (i.e., $\mathsf{Com}_{\mathsf{NM}}$ session with V_ℓ). However, if the first message of $\mathsf{Com}_{\mathsf{NM}}$ is not the determining message, the case analyzed should include the possibility that the commitment to be exposed has already started before FM_i occurs, but has not reached its determining message yet. Then we modify M'_ℓ to be initialized to the point where either the $\mathsf{Com}_{\mathsf{NM}}$ session with V_ℓ starts or where FM_i occurs in the main thread, whichever occurs first. If the former occurs first, M'_ℓ needs to carry out the execution of the look-ahead threads *until it reaches* FM_i. However since it cannot rewind the external receiver, in the look-ahead threads it must internally simulate the receiver. This is similar to the situation faced in creating the standalone adversary for extraction in the final stage of building $\mathcal{S}$. Indeed, the condition we used there, namely that the receiver has no private coin until the knowledge-determining message, implies that the receiver has no private coins until the point where the simulation reaches FM_i (because it occurs before the determining message, which in turn occurs before the KDM). Hence M'_ℓ can carry out the simulation of the look-ahead threads internally until it reaches FM_i.

3 Impossibility result for concurrent non-malleable general functionalities.

In this section we show that it is *impossible* to extend the result we achieved for zero knowledge for general functionalities. Specifically, we will 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.

The function $\mathcal{F}$. The function $\mathcal{F}$ will be a combination of $\binom{2}{1}$ string oblivious transfer and zero knowledge for a particular language. This is a two-party functionality where only one party (which we call the *receiver*) gets any output. Formally, it is defined as follows:

[11] There is also a notational difference: we defined $p(i)$ to be the index of the prover *sending* FM_i. Now, depending on whether FM_i is from the committing party or from the receiver, $p(i)$ will be the index of the prover sending *or* receiving FM_i, respectively.

The functionality will be parameterized with a security parameter n. Let $f : \{0,1\}^n \rightarrow \{0,1\}^n$ be a one-way function, define

$$\mathcal{F}_{ZK}(w \circ x, x) = \begin{cases} 1 & x = f(w) \\ 0 & \text{otherwise} \end{cases}$$

For $x, w \in \{0,1\}^n$, and where $\circ$ denotes concatenation. That is $\mathcal{F}_{ZK}$ is the ideal zero knowledge functionality for the $\mathbf{NP}$-relation $R_f = \{(x, w) : x = f(w)\}$.

We define $\mathcal{F}_{OT}$ as follows

$$\mathcal{F}_{OT}(x_0 \circ x_1, b) = x_b$$

For $x_0, x_1 \in \{0,1\}^n$ and $b \in \{0,1\}$. That is, $\mathcal{F}_{OT}$ is the functionality for $\binom{2}{1}$ string oblivious transfer, where the sender has two strings as inputs $x_0, x_1 \in \{0,1\}^n$, the receiver one bit $b \in \{0,1\}$, the receiver learns x_b but not x_{1-b} and the sender learns nothing about b.

We define $\mathcal{F}$ to be the function that allows to compute both $\mathcal{F}_{ZK}$ and $\mathcal{F}_{OT}$. Formally, it is defined as follows:

$$\mathcal{F}(i \circ x \circ w \circ x_0 \circ x_1, i' \circ x \circ b) = \begin{cases} \mathcal{F}_{ZK}(x \circ w, x) & i = i' = 0 \\ \mathcal{F}_{OT}(x_0 \circ x_1, b) & i = i' = 1 \\ \perp & \text{otherwise}(i \neq i') \end{cases}$$

(where $\perp$ is a value denoting failure).

Let Π be a two-party protocol, where we call one party the sender and the other the receiver. We say that Π computes $\mathcal{F}$ if when both parties follow the protocol's instructions with inputs $\mathbf{s}$ and $\mathbf{r}$ respectively, the receiver outputs $\mathcal{F}(\mathbf{s}, \mathbf{r})$. (To simplify notations, we assume that the value n is used as the security parameter of the protocol Π as well as the input size parameter of the functionality.) We prove the following theorem:

Theorem 3.1. *Assume that f is a one-way function f and let $\mathcal{F}$ be defined as above. Let Π be any polynomial-time two party protocol computing $\mathcal{F}$. Then, there's a polynomial $t(\cdot)$ such that for any n there exists a distribution D_Π on $t = t(n)$ input pairs for Π, a setting of one party in each of t pairs as corrupt, and a polynomial function* SECRET *that maps the inputs into $\{0,1\}^n$, so that the following hold.*

- *There exists a polynomial-time adversary A which controls the corrupt parties (including the schedule and contents of those parties' messages) such that in a concurrent execution of t copies of Π, with the parties receiving inputs chosen from the distribution D_Π, the adversary A outputs the value of* SECRET *on the inputs with probability* 1.

- *For any polynomial-time adversary $\hat{A}$ which controls the corrupt parties in an ideal model execution wherein the parties get access to t copies of the ideal functionality $\mathcal{F}$, and receive inputs chosen from the distribution D_Π, the probability that $\hat{A}$ outputs the value of* SECRET *on the inputs is negligible, the probability taken over D_Π and the coins of $\hat{A}$.*

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.

This result is somewhat surprising since in many previous settings, (UC-security [CLOS02], bounded composition [Lin03a, Pas04], timing [KLP05], super-polynomial simulation [PS04, BS05]) obtaining a composable zero knowledge protocol implied obtaining a composable protocol for general functionalities. In fact, there is a natural candidate for such a protocol in the case of oblivious transfer: to transform the Naor-Pinkas OT protocol [NP99] to handle malicious adversaries we only need one application of zero-knowledge proofs, and in that application the receiver proves a statement that is independent of any messages sent to it by the prover (and hence can be thought of as secret input that is fixed in advance). Thus, it may seem that combining this protocol with the zero knowledge argument of Section 2 would yield an implementation of $\mathcal{F}$. In fact, it may seem that by combining Naor-Pinkas OT with our zero knowledge and Yao's garbled circuit protocol [Yao86], we might get a protocol for computing *any* deterministic function assuming that the inputs are fixed in advance, this is because the compiler to ensure security against malicious adversaries, no party ever needs to use zero knowledge to prove statements that depend on the messages sent by the other party. Thus, it may seem that we don't need zero knowledge for adaptively chosen inputs in this case.

However, it turns out this is not the case. The problem in proving the security of this particular protocol is that when performing the simulation and rewinding the zero-knowledge protocol, we may also rewind other executions of the OT protocol, which is problematic in the case of an honest sender (as the security of the OT requires that the receiver will only learn one of the sender's inputs). Indeed, by the results of Lindell [Lin04], no *black-box* simulator can work in this case. Nonetheless, when general composition with arbitrary protocols is not required, the fact that the straightforward black-box simulation does not work does not mean that there's no other more clever simulation to prove the security of this protocol. The results of this section will rule out this possibility as well.

The proof of Theorem 3.1 will proceed in two stages:

1 First, (as warm-up) we will prove that for every protocol Π_{ZK} for the zero knowledge functionality (for the relation R_f above), there exists an ideal two-party non-interactive deterministic functionality 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. (In the same sense as Theorem 3.1, that for inputs chosen from some distribution and fixed in advance, an adversary can learn a secret that she cannot learn if Π_{ZK} was replaced with the ideal zero knowledge functionality.)

   Note that the inputs to F_Π are not adaptively chosen. For the second stage we would require F_Π to be non-interactive (i.e., not *reactive*). If F_Π were allowed to be a reactive functionality or use adaptively chosen inputs, then this would be the same setting as the results for impossibility of protocols that are secure under general composition or the "chosen protocol attack." That is, the results of [Lin03b] (and in fact implicitly earlier works such as [Can01, CF01, CKL03, KSW97]) imply that for every zero knowledge protocol Π_{ZK}, we can find a *protocol* P (depending on Π_{ZK}) such that the concurrent execution of Π_{ZK} and P is insecure in the above sense. 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.

   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.

2 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 [YAO86]. 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 [BGI+01]'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 F_Π 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 succeed in obtaining a secret, whereas in the ideal world (where Π_{ZK} is not being executed), no adversary can succeed.

3.1 Proof of Theorem 3.1: First stage.

We now prove the following lemma (this is the formalization of Item 1 from above):

Lemma 3.2. *Suppose that f is a one-way function and let R_f be the $\mathbf{NP}$-relation $\{(x,w) : x = f(w)\}$. Let Π be a stand-alone zero-knowledge proof of knowledge for the relation R_f with $k = k(n)$ prover messages (where n denotes both the security parameter and the length $|x|$ of the statement being proven). Then, there exists a polynomial-time function $F = F_\Pi : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$, a distribution D on $(\{0,1\}^*)^{k+1}$, a function* SECRET $: (\{0,1\}^*)^{k+1} \to \{0,1\}^n$ *and a polynomial-time adversary A such that:*

- *In a concurrent execution scheduled by A of one copy of Π, with A as verifier, and k ideals calls to F with A providing the second input and receiving the output, if the inputs to the honest parties are $\mathbf{d}$ chosen from D, then A learns* SECRET$(\mathbf{d})$ *with probability one.*
- *In any execution of k copies of the ideal calls to F and a copy of the ideal functionality $\mathcal{F}_{ZK}$, with honest inputs $\mathbf{d}$ chosen from D, a polynomial time adversary $\hat{A}$ will only output* SECRET$(\mathbf{d})$ *with negligible probability.*

Proof. (Sketch) Before proving the lemma, let us recall why there do not exist (in the plain model) protocols for zero-knowledge (proof of knowledge) that are secure against a *chosen protocol attack*. Let Π be a standalone zero knowledge proof of knowledge protocol. Think of the following scenario involving four parties Alice, Bob, Charlie and David: there's a public value x and both Alice and David share a secret value w such that $x = f(w)$. We consider two simultaneous executions: in one execution Alice will prove to Bob that she knows such a value w using the ZKPOK protocol Π. In the second execution Charlie and David will run the protocol Π' defined as follows: At first Charlie will prove to David that he knows such a value w using the protocol Π.[12] Then, if this succeeds, David will send w to Charlie.

It's clear that if Bob and Charlie are coordinating a malicious attack, then they can learn the value w. However, if the execution of Π was replaced with an ideal call to the functionality $\mathcal{F}_{ZK}$, then it does not help the adversary in executing Π. In that case, since Π is a proof of knowledge and f is a one-way function, it follows that Charlie would not be able to run Π with David, and so will not learn the value w. This is basically the proof that there's no zero knowledge proof of knowledge protocol Π that is secure under general composition/chosen protocol attack.

[12]If the protocol Π refers to the identities of the parties, we define that when executing this internal copy of Π, Charlie will use the identity "Alice" and David will use the identity "Bob". Note that we're free to define Π' to depend on Π in an arbitrary way.

We now want to convert Π' from a protocol into k ideal calls to a functionality F which uses inputs that are chosen from some distribution and fixed in advance. The natural thing is to simply use for F the *next message function* of David's strategy in the protocol Π'. That is, the inputs will be w, a string r that is chosen at random, and on input a transcript $\mathbf{t} = \langle d_1, c_1, d_2, c_2, \ldots, d_i, c_i \rangle$ of Charlie and David's messages in the first i rounds of Π', the function F will output David's $i + 1^{th}$ message in this protocol given that his input is w, his random coins are r, and the transcript until that point was $\mathbf{t}$. If we use this F then certainly in a coordinated attack, Bob and Charlie can emulate the attack above and learn the value w. However, it's not at all clear that this is not possible in the ideal world as well— indeed if Π is a black-box zero knowledge proof, given the ability to query the next-message function of David one can certainly obtain an accepting transcript where David is the verifier.

To make the attack infeasible in the ideal model, we add to the inputs a key s for a message authentication (MAC) scheme. Now, given such a transcript $\mathbf{t} = \langle d_1, c_1, d_2, c_2, \ldots, d_i, c_i \rangle$, the function F will request also a valid tag/signature (with respect to the key s) on the prefix $\langle d_1, c_1, d_2, c_2, \ldots, d_i \rangle$, and will output not only d_{i+1} but also a tag on $\langle d_1, c_1, d_2, c_2, \ldots, d_i, c_i, d_{i+1} \rangle$. It's not hard to see that now, given only k queries to F, it's infeasible for a polynomial-time adversary to obtain w without essentially interacting with Π' in a straightline manner (i.e., submitting k queries of increasing and consistent transcripts). Thus, in this ideal world, the soundness/proof of knowledge property of Π implies that the probability that an adversary outputs w is negligible. $\square$

3.2 Proof of Theorem 3.1: Second stage.

We will now finish the proof of Theorem 3.1. As mentioned above, the idea would be to "compile" the scenario of the first stage into a scenario where, in the real model, the only protocol executed is the oblivious transfer protocol (apart from one execution of the ZKPOK protocol Π_{ZK}). We do this using a modification of Yao's "garbled circuit" method following the intuition given above. We note that we will not be using Lemma 3.2 as a black-box but rather will follow the proof of this lemma to prove the theorem.

Yao's garbled circuit technique. We now sketch Yao's method. As this method is well known we focus on our notations and particular conventions. See [LP04] (whose notations we follow) for a full description of the method and its analysis. We will have two parties, a *sender* and a *receiver*. Let n be a security parameter (we will use $\binom{2}{1}$ string OT for strings of length $2n$). The sender holds a circuit C (where $|C|$ is of some polynomial size, and this size and the topological structure of the circuit are not secret), and the receiver holds an input x. The goal is for the receiver to learn $C(x)$ but nothing else about the circuit C. For every wire w in the circuit and bit $\sigma \in \{0,1\}$ we define a value k_w^σ which is chosen uniformly at random from $\{0,1\}^n$. The garbled circuit consists of tables that allow you for any gate g (where $g : \{0,1\} \times \{0,1\} \to \{0,1\}$) that takes input wires w_1, w_2 and has one output wire w_3, to compute $k_{w_3}^{g(\sigma_1,\sigma_2)}$ from $k_{w_1}^{\sigma_1}$ and $k_{w_2}^{\sigma_2}$. The table is obtained by taking a private-key CPA-secure encryption scheme, and having for each gate g a table with four rows: for every $\sigma_1, \sigma_2 \in \{0,1\}$ we place the encryption of $k_{w_3}^{g(\sigma_1,\sigma_2)} \circ 0^n$ with the key $k_{w_1}^{\sigma_1}$ and then with the key $k_{w_2}^{\sigma_2}$ (where $\circ$ denotes concatenation).[13]

Typically, for every output wire w of the circuit, one also supplies a way to compute σ from k_w^σ. That is, for the tables corresponding to output gates, the value σ (appropriately padded) is encrypted instead of the value k_w^σ. However, we will do something slightly different: we will XOR the output with some secret string $z \in \{0,1\}^m$ (where m denotes the number of outputs). That is, we will encrypt in these tables the

[13]The reason for padding with zeros is to make sure that when trying to decrypt with the two keys all rows in the table, the receiver will know when it found the right row.

value $\sigma \oplus z_w$, where w is the label of that output wire. We will choose the string z to be $G(s)$ where $G : \{0,1\}^n \to \{0,1\}^m$ is a pseudorandom generator and s is chosen uniformly at random from $\{0,1\}^n$.

The Yao protocol. The protocol is typically as follows: the sender sends the garbled circuit over to the receiver, but keeps to itself the keys corresponding to each of the input wires. Then, by performing m' executions of string $\binom{2}{1}$ OT (where m' is the number of input wires), for each input wire w the receiver chooses to get either k_w^0 or k_w^1, according to the value in the w^{th} position of its input x. We will make the following changes:

1 Instead of sending the garbled circuit to the receiver, we will assume that the garbled circuit is an input that is given to the receiver. We note that we will always have the receiver as a corrupted party. Thus, we think of the scenario where the inputs are chosen from a distribution (not a product distribution), and these inputs are given to both the honest parties and corrupted parties. This distribution will provide the honest parties with the keys for the input wires, and the corrupted parties with the corresponding garbled circuit. Note that this means that there's no issue of trust that the circuit is indeed garbled correctly.

2 We will select $s_1, \ldots, s_{m'}$ uniformly at random from $\{0,1\}^n$ subject to the condition $s_1 \oplus s_2 \oplus \cdots \oplus s_{m'} = s$ (recall that the "mask" to the outputs is $z = G(s)$). In the OT, for every input wire w, the sender will use as the two input strings $k_w^0 \circ s_w$ and $k_w^1 \circ s_w$. The idea is that before concluding *all* the copies of the OT corresponding to this circuit, the receiver will not get any information about the output. On the other hand, we note that by [Yao86] (see [LP04] for details), once a corrupted party finishes all OT's needed to obtain input strings corresponding to its chosen input a, it will learn only the output of $C(x)$, and nothing more.

Note that this means that the only interaction between the sender and receiver is performing the m' copies of the OT.

Our compiler. Let Π be a protocol that implements $\mathcal{F}$. We can derive from Π protocols Π_{ZK} and Π_{OT} for zero-knowledge (for the relation R_f) and oblivious transfer, respectively, by simply having the sender and receiver choose the appropriate value of i. From Π_{ZK}, let $F = F_{\Pi_{ZK}}$ be the function obtained from the proof of Lemma 3.2. If k is the number of prover messages for Π_{ZK}, and m is the length of the input to F, we now compile the k copies of F from the proof of Lemma 3.2 following the procedure above into km copies of the OT functionality (which is a subfunctionality of $\mathcal{F}$) with inputs as above. Consider an execution of one copy of Π_{ZK} (with inputs $x = f(w)$ for w chosen at random) concurrently with these km copies of Π_{OT} (or equivalently, execution of $km + 1$ copies of Π) where the corrupt parties are the receivers in all cases (i.e., the verifier in the zero-knowledge, and the receiver in the OT). We make the following claims:

- In the real world, there is an adversary which can learn w with probability one.

  This follows by combining the adversary strategy given in the proof of Lemma 3.2 – in which the adversary needs access to k evaluations of the F functionality to learn w – with the Yao protocol – which exactly allows the adversary to evaluate the F functionality. Thus, this scenario allows the adversary to obtain the value w with probability one.

- In the ideal world, no adversary can learn w with non-negligible probability.

  In the ideal world the adversary basically gets one call to the ZK functionality $\mathcal{F}_{ZK}$ and km calls to the OT functionality $\mathcal{F}_{OT}$ (with honest parties' inputs chosen as described above) – note that since both parties must agree on how to use $\mathcal{F}$, if the adversary tries to use $\mathcal{F}_{ZK}$ more than once, or $\mathcal{F}_{OT}$ $km + 1$ times, then this will result in the adversary getting the output $\perp$. The adversary gains no information (in an information-theoretic sense) about w from its one interaction with $\mathcal{F}_{ZK}$. However, it's more tricky to show that it won't learn anything from the OT calls.

  The adversary has access to km copies of $\mathcal{F}_{OT}$, and we divide these copies to sets $S_1, \ldots, S_k$, where $|S_i| = m$ for all i, and contains all copies of $\mathcal{F}_{OT}$ corresponding to a single garbled circuit. From the proof of security of Yao's protocol (see [LP04] for details), we can show that the adversary gets no information (in a complexity-theoretic sense) about the circuit except for its value on the outputs corresponding to the adversaries choice as a receiver in the OT executions. Furthermore, because of the secret-shared "mask" we use, before the adversary queries *all* the copies of S_i she gets no information about the output of the i^{th} circuit. Assume the sets $S_1, \ldots, S_k$ are ordered according to the timing of the query to the last copy of the OT in each set S_i. We can simulate the adversary by an adversary in the model where all the invocations in the set S_i are replaced with one invocation to the functionality F (the simulator will provide random answers until the last query). However, this is exactly the model of Lemma 3.2 and so, as in that case, the adversary will only learn w with negligible probability.

$\square$

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 [LIN03B, LIN04], 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 [PS04], 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 one could derive results such as [CLOS02, BS05] and our current result by just plugging in the appropriate functionality.

References

[BAR01] B. Barak. How to go beyond the black-box simulation barrier. In *Proc.* 42*nd FOCS*, pages 106–115, 2001.

[BAR02] 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.

[BCNP04] 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.

[BGI+01] 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.

[BS05] 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.

[BOGW88] 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.

[BLU87] M. Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, pages 1444–1451, 1987.

[CAN01] 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.

[CAN05] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. Cryptology ePrint Archive, Report 2000/067, 2005. Revised version of [CAN01].

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

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

[CKL03] R. Canetti, E. Kushilevitz, and Y. Lindell. On the Limitations of Universally Composable Two-Party Computation Without Set-up Assumptions. In *Eurocrypt '03*, 2003.

[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally Composable Two-party Computation. In *Proc.* 34*th STOC*, pages 494–503, 2002.

[DPP97] 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.

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

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

[GM00] J. A. Garay and P. D. MacKenzie. Concurrent Oblivious Transfer. In *Proc.* 41*st FOCS*, pages 314–324, 2000.

[GOL01] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001. Earlier version available on `http://www.wisdom.weizmann.ac.il/~oded/frag.html`.

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

[GMR84] 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.

[HHK+05] 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.

[HM96] S. Halevi and S. Micali. Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing. In *CRYPTO*, pages 201–215, 1996.

[KLP05] 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.

[KSW97] 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.

[KP01] J. Kilian and E. Petrank. Concurrent and resettable zero-knowledge in poly-logarithm rounds. In *Proc. 33th STOC*, pages 560–569, 2001. Preliminary full version published as cryptology ePrint report 2000/013.

[LP04] Lindell and Pinkas. A Proof of Yao's Protocol for Secure Two-Party Computation. In *ECCCTR: Electronic Colloquium on Computational Complexity, technical reports*, 2004.

[LIN03A] Y. Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *Proc. 35th STOC*, pages 683–692, 2003.

[LIN03B] Y. Lindell. General Composition and Universal Composability in Secure Multi-Party Computation. In *Proc. 44th FOCS*, pages 394–403, 2003.

[LIN04] Y. Lindell. Lower Bounds for Concurrent Self Composition. In *Theory of Cryptography Conference (TCC)*, volume 1, pages 203–222, 2004.

[MMY06] T. Malkin, R. Moriarty, and N. Yakovenko. Generalized Environmental Security from Number Theoretic Assumptions. In *TCC '05*, 2006.

[NAO89] M. Naor. Bit Commitment Using Pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991. Preliminary version in CRYPTO' 89.

[NOVY98] 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.

[NP99] M. Naor and B. Pinkas. Oblivious Transfer with Adaptive Queries. In *Crypto '99*, pages 573–590, 1999.

[PAS03] R. Pass. Simulation in Quasi-Polynomial Time, and Its Application to Protocol Composition. In *Eurocrypt '03*, 2003.

[PAS04] R. Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *Proc. 36th STOC*, pages 232–241, 2004.

[PR03] R. Pass and A. Rosen. Bounded-Concurrent Secure Two-Party Computation in a Constant Number of Rounds. In *Proc. 44th FOCS*, 2003.

[PR05A] R. Pass and A. Rosen. Concurrent Non-Malleable Commitments. In *Proc. 46th FOCS*, 2005.

[PR05B] R. Pass and A. Rosen. New and Improved Constructions of Non-Malleable Cryptographic Protocols. In *Proc. 37th STOC*, 2005.

[PRA05] M. Prabhakaran. *New Notions of Security*. PhD thesis, Department of Computer Science, Princeton University, Princeton, NJ, USA, 2005.

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

[PS04] M. Prabhakaran and A. Sahai. New notions of security: achieving universal composability without trusted setup. In *Proc. 36th STOC*, pages 242–251, 2004.

[RK99] R. Richardson and J. Kilian. On the Concurrent Composition of Zero-Knowledge Proofs. In *Eurocrypt '99*, pages 415–432, 1999.

[SAH99] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *Proc. 40th FOCS*, pages 543–553, 1999.

[YAO86] A. C. Yao. How to Generate and Exchange Secrets. In *Proc. 27th FOCS*, pages 162–167, 1986.